Experimental Modal Analysis and Model Validation of Antenna Structures

by

Brendon Ryan Potgieter

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Mechanical Engineering at Stellenbosch University

Department of Mechanical and Mechatronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Prof G. Venter

December 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

Experimental Modal Analysis and Model Validation of Antenna Structures

B R Potgieter

Department of Mechanical and Mechatronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Mech)

December 2010

Numerical design optimisation is a powerful tool that can be used by engineers during any stage of the design process. Structural design optimisation is a specialised usage of numerical design optimisation that has been adapted to cater specifically for structural design problems. A specific application of structural design optimisation that will be discussed in the following report is experimental data matching. Data obtained from tests on a physical structure will be matched with data from a numerical model of that same structure. The data of interest will be the dynamic characteristics of an antenna structure, focusing on the mode shapes and modal frequencies. The structure used was a scaled, simplified model of the Karoo Array Telescope-7 (KAT-7) antenna structure.

Experimental data matching is traditionally a difficult and time-consuming task. This report illustrates how optimisation can assist an engineer in the process of correlating a finite element model with vibration test data.

Uittreksel

Eksperimentele Modale Analise en Modelbekragtiging van Antennastrukture

B R Potgieter

Departement Meganiese en Megatroniese Ingenierswese
Universiteit van Stellenbosch
Privaat Sak X1, 7602 Matieland, Suid-Afrika

Tesis: MScIng (Meg)

Desember 2010

Numeriese ontwerp-optimisering is 'n kragtige ingenieurshulpmiddel wat tydens enige stadium in die ontwerpsproses ingespan kan word. Strukturele ontwerp-optimisering is 'n gespesialiseerde gebruik van numeriese ontwerp-optimisering wat aangepas is om spesifiek van diens te wees by die oplos van strukturele ontwerpsprobleme. 'n Spesifieke toepassing van strukturele ontwerp-optimisering wat in hierdie verslag bespreek sal word, is eksperimentele datakorrelasie. Data afkomstig van toetse op 'n fisiese struktuur sal gekorreleer word met data afkomstig van 'n numeriese model van die selfde struktuur. Die data van belang is die dinamiese eienskappe van 'n antennastruktuur, spesifiek die modusvorme en modale frekwensies. Die betrokke struktuur wat gebruik is, is 'n vereenvoudigde skaalmodel van die Karoo Array Telescope-7 (KAT-7) antennastruktuur.

Eksperimentele datakorrelasie is, tradisioneel gesproke, 'n moeilike en tydrowende taak. Hierdie verslag sal illustreer op watter wyse optimisering 'n ingenieur van hulp kan wees in die proses om 'n eindige elementmodel met vibrasietoetsdata te korreleer.

Acknowledgements

I would like to thank the following persons for their contributions toward this project:

- Prof. G Venter for his guidance throughout the project.
- The team in the workshop of the Department of Mechanical Engineering for their help in the manufacturing and assembling of the physical model.
- Mr. JJ Heunis for his help during the testing phase of the project.
- To SKA for the funding for this project and giving me the opportuinty to work on this project.
- Miss A Perold for her help in the editing of the written text.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the 20^{th} century, telescopes revealed an expanding universe with billions of galaxies, all of which are filled with stars of various sizes and temperatures, along with black holes, neutron stars, planets and gas clouds (Ready to host the SKA, 2008). Now in the 21^{st} century, gaining an understanding of the universe and its contents is the next challenge. For this challenge, a new generation of astronomical facilities is required. The Square Kilometre Array (SKA), a revolutionary radio telescope, will play a crucial role in the attainment of the aforementioned. South Africa and Australia are the two countries short listed to host the SKA, which will be the most powerful radio telescope ever constructed (SKA Project [Online]).

The SKA project will consist of about 3000 antennas with a diameter of 10 to 15 meters each. Half of the antennas will be concentrated in a 5 km diameter central region, and the rest distributed as far as 3000 km from the central region. The distribution region is shown in Fig. 1.1. These antennas will collectively form a radio telescope with an extremely large collecting area, allowing it to be 50 times more sensitive and able to survey the sky 10000 times faster than any radio telescope array built before (Ready to host the SKA, 2008).

Figure 1.1: Distribution region of the SKA antennas (Ready to host the SKA, 2008)

The SKA will be used to pick up radio waves from pulsars, quasars, masers, distant galaxies and other objects in the universe. The information gained from these radio waves will enable radio astronomers to study phenomena such as the birth and death of stars. The radio waves that eventually reach Earth are, however, very weak due to the immense distances that they travel. The consequent need for a large collection area is what has driven the design of an array of radio telescopes.

In order to illustrate their commitment to the project, as well as their possession of the necessary skills to host the SKA, South Africa has started with a smaller demonstration project called the MeerKAT telescope. The MeerKAT will furthermore contribute to the development of the technology and science of the SKA. Phase 1, the KAT-7, will consist of the first seven 12 m diameter antennas of the MeerKAT that will, in turn, eventually consist of 80 antennas (MeerKAT Construction [Online]).

1.2 Scope

This report will illustrate the value of numerical design optimisation in engineering design processes. According to Vanderplaats (2007), much of an engineering design task is quantifiable, enabling the engineer to employ a computer in the rapid analysis of alternative designs. The purpose of numerical optimisation is to search for the design that best meets the design requirements. Numerical design optimisation performs iterative tasks, leaving the engineer free to concentrate on providing the correct input as well as evaluating and interpreting the output. It furthermore provides increased insight into a problem and reduces design time.

Structural design optimisation is a specialised utilisation of numerical design optimisation that has been adapted to cater specifically for structural de-

sign problems. Some of the most common structural optimisation applications include the mass minimisation of a structure, concept generation, concept evaluation, structure failure prevention and data matching. This report focuses on the specific application of experimental data matching. The data of interest is the dynamic characteristics, specifically the modal frequencies and mode shapes, of an antenna structure.

Mundt and Quinn (2005) describe the correlation of test data with a finite element (FE) model as a complex and arduous task, traditionally performed manually. Such data correlation or matching typically requires a comparison of the numerical and test data. On the basis of this comparison, an assessment is made of which parameters need to be changed and to what degree. The aforementioned changes are next made by manually editing the input data and rerunning the FE analysis. This iterative process continues until the analyst has tuned the FE model to be as accurate a representation of the physical model as possible, within the constraints of time and funding. This tuning is essentially performed one mode at a time. Due to the difficulties associated with experimental data matching, only simplified FE models are used for correlation. Detailed FE models make the task too complicated for a designer to perform manually.

A simplified, scaled model of an antenna structure was designed and manufactured in order to provide a physical structure for vibration testing. A FE model of the physical model was also generated, thereby providing two models of which the data needed to be matched.

The antenna structure considered in this study was a simplified, scaled model of the Karoo Array Telescope-7 (KAT-7) currently being constructed in South Africa as part of its bid for the SKA project. This model was used in order to perform vibration tests which provided a set of frequencies and mode shapes. A FE model of the antenna was generated and then subjected to optimisation techniques in order to match the numerical data with the test data. *GENESIS*, a structural optimisation program, was used to apply the optimisation techniques.

1.3 Motivation

The purpose of this project is to illustrate the manner in which optimisation can assist an engineer faced with an experimental data matching problem. Firstly, a number of the reasons for considering optimisation will be mentioned. As already mentioned experimental data matching with a FE model can be a challenging exercise. Due to the difficulties associated with experimental data matching, in the past, analysts have been limited to very simplified FE models. If models were too detailed then the time and cost involved in matching the data become too great to make it sensible for the analyst to complete. Due to this the data matching process is often stopped when a project has run out of time or money - not necessarily when the data was perfectly matched. Optimisation allows for the automation of the matching process that in turn allows for larger and more detailed models to be analysed.

The main advantage of using optimisation is that it allows for a more accurate FE model - an important function taking into consideration that analysts rely on FE models to provide them with a better understanding of a structure's response. The response of interest for this project is that of an antenna, which collects radio wave signals, to environmental conditions. Understanding the structure's dynamic behaviour will enable an analyst to ensure that the antenna is indeed collecting the signal correctly.

1.4 Objectives

The main objective of this project is to illustrate how optimisation can be used to gain a FE model that can accurately model the dynamic characteristics of a physical model. This will be achieved by completing the following steps:

- Designing and manufacturing a simplified, scaled model of the KAT-7 antenna structure
- Generating a FE model of the physical model
- Conducting vibration tests on the physical model in order to gain its dynamic characteristics
- Matching the physical data with the numerical data by using optimisation techniques

- Verifying the accuracy of the matched model by using different boundary conditions

1.5 Report Outline

The current chapter provides a short overview of the main aspects of the project. These aspects include an introduction to optimisation as well as the origin of the project, i.e. the KAT-7. The motivation for and the objectives of the project are also defined.

In Chapter 2, a detailed literature review of antenna structures, experimental modal analysis, data matching and optimisation is presented.

Chapter 3 deals with the physical model used in this project by providing a detailed discussion of the design and manufacturing of this simplified, scaled model. Detail of the KAT-7, on which the model is based, is also given.

In Chapter 4, the FE models and FE analysis are discussed. An overview of the analysis done on the KAT-7 is given. This is followed by information on the generation and analysis of the simplified model's FE model, as well as the results thereof.

Chapter 5 presents the detail of the vibration tests performed on the physical model. This includes information on the experimental set-up, the procedure and the results obtained from the tests.

Chapter 6 provides a discussion of the manner in which the test data and the numerical data were compared.

In Chapter 7, the optimisation techniques used in the project are discussed. It also covers how the optimisation problem was formulated and how that formulation was arrived at. The results of the optimisation process are also provided.

In Chapter 8, the validation of the matching process is discussed by performing a data comparison of the two models with different boundary conditions.

Finally, in Chapter 9, final conclusions and a few suggestions for future work are provided.

Chapter 2

Literature Study

A study of relevant literature was conducted in order to gain a better understanding of the main components involved in this project. Some such components discussed here include antenna structures, experimental modal analysis, data matching and optimisation.

2.1 Antenna Structures

A radio telescope is an extremely sensitive radio wave receiver consisting of two basic components, namely the dish and the receiver or feed. The dish collects and reflects radio waves to the feed, which serves as the central focal point. The radio signal is then amplified and digitised in order to be stored and analysed by a computer. This process is illustrated in Fig. 2.1.

The dish and feed collectively will henceforth be referred to as the antenna. The dynamics of the antenna are of significant interest as it determines the accuracy with which the radio waves are reflected to the focal point. The aforementioned accuracy in turn determines the overall quality of the received signal.

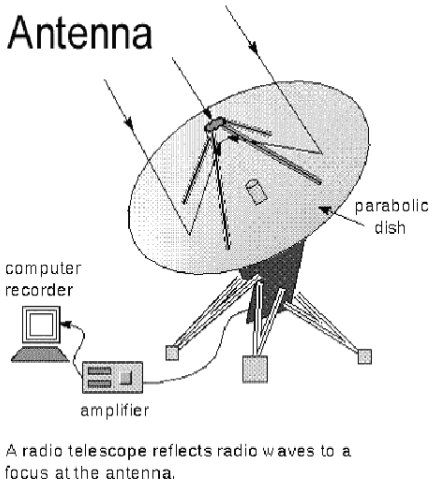


Figure 2.1: Reflecting dish (Haystack Radio Telescope [Online])

2.2 Dynamic Characteristics

The dynamic characteristics of a structure can be organised into three related categories. Ewins (1984) terms these categories the "spatial model", "modal model" and "response model".

The spatial model is essentially a description of a structure's physical characteristics, as determined by mass, stiffness and damping properties.

The modal model is a description of the manner in which a structure will vibrate naturally in the case of no external excitation occurring. This description is provided in the form of a set of vibration modes that includes natural frequencies with corresponding vibration mode shapes and modal damping factors. The modes and frequencies are consequently referred to as "natural" modes or frequencies.

The response model is a description of the manner in which a structure will vibrate under external excitation. The latter vibration will depend on the structure's inherent properties as well as the nature and magnitude of the excitation.

It is important to understand how the above three models are related, refer Fig. 2.2 and Fig. 2.3. The properties of the spatial model give rise to the modal model, which in turn gives rise to a response model. The response model is furthermore dependent on the excitation input. The order in which the three models are generated can be reversed, although it would limit the comprehensiveness of the spatial model.

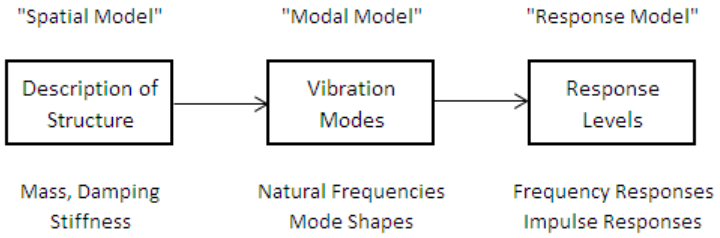


Figure 2.2: Theoretical route to vibration analysis (Ewins, 1984)

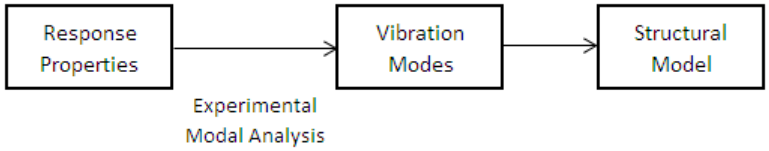


Figure 2.3: Experimental route to vibration analysis (Ewins, 1984)

2.3 Vibration Testing

Experimental modal analysis, in essence, involves the extraction of modal data from a physical structure by conducting vibration tests. The following section provides a discussion of a number of the aspects to be considered during vibration testing, namely the manner in which to support and excite the structure as well as the manner in which to obtain the required data from the vibration tests.

2.3.1 Supporting the Structure

The two types of support used during vibration testing, namely soft and grounded supports, are discussed below.

Soft Supports

Ewins (1984) states that soft supports are used to model 'free-free' support conditions. In the case of 'free-free' support conditions, the test structure is not attached to the ground at any of its coordinates and is, in effect, freely suspended in space. Under these conditions, the structure will have 6 rigid body modes that are determined solely by its mass and inertia properties and which do not contain any deformation modes. These rigid body modes are all at a frequency of 0 Hz.

In practice, of course, 'free-free' support conditions are not possible, as the structure must be supported in some way or another. These conditions can be approximated by using a suspension system, e.g. hanging the structure from bungy cords or resting it on a tire tube. The suspension system will cause the rigid body modes' frequencies to become non-zero, although they will still be very low in relation to the deformation modes of the structure. Ewins (1984) therefore states that the highest rigid body mode's frequency must be less than 10-20 % of that of the lowest deformation mode.

Ewins (1984) furthermore states that a rigid body will possess 6 rigid body modes and that it is necessary to check that the natural frequencies of all of these are sufficiently low before being satisfied that the suspension system is sufficiently soft.

Grounded Supports

Grounded support involves the fixing of a structure to a ground support or to the ground itself. This support needs to act as a completely rigid structure in order to prevent it from affecting the test structure's dynamic characteristics. Ensuring such a completely rigid structure is, however, complex in practice as finding a base or foundation for the test structure that will be sufficiently rigid to provide the necessary degree of grounding is difficult (Ewins, 1984). Ewins therefore suggests measuring the mobility of the base structure itself over the test's frequency range and ensuring that the aforementioned is much lower than the corresponding levels for the test structure at the point of attachment.

Another aspect for consideration is local stiffening at the grounded attachment. A check for the repeatability of results can be performed by disassembling and then reassembling the grounding connection.

2.3.2 Excitation of the Structure

The excitation of a structure can be achieved in a variety of ways, using a variety of devices, depending on the desired type of excitation. The relevant devices, termed 'exciters', can be of either a mechanical or, more commonly, electromagnetic nature. By employing exciters a continuous sinusoidal, random or periodic excitation can be induced onto the structure. Care needs to be taken in its set-up and in the manner in which it is attached to the structure. As stated by Ewins (1984) a structure will not only respond in the same direction as the excitation but may respond in others as well. An exciter is generally very stiff in the directions that are not in its driving direction. Due to this if a structure responds in a direction other than the driving direction the exciter will resist this response and in effect introduce a secondary excitation (Ewins, 1984). This problem is overcome by attaching the exciter to a structure through a driving rod or 'stinger'. The stinger has the characteristic of being stiff in one direction (excitation direction) while being flexible in the other five directions (Ewins, 1984). The stinger should not be too long or flexible as it may then introduce its own resonances into the measurements. As exciters are attached to a structure, Inman (2001) warns that a set-up should be done in such a way that the exciter's own mass is not introduced into the system. The exciter should furthermore be perpendicular to the direction of the intended excitation.

The use of a modal hammer is a further means of exciting frequencies in a structure. The excitation device consists of a hammer with a force transducer built into its head (Inman, 2001). The aim in using this device is to apply an impulse to the structure which will excite the system's natural frequencies (Inman, 2001). Inman states that the peak impact force is almost proportional to the hammerhead's mass and the impact velocity. Different hammerheads allow the excitation of different frequency ranges for structures with different degrees of stiffness. Using a hammer does, however, pose certain difficulties such as a lack of control over the exact impact zone, orientation and force.

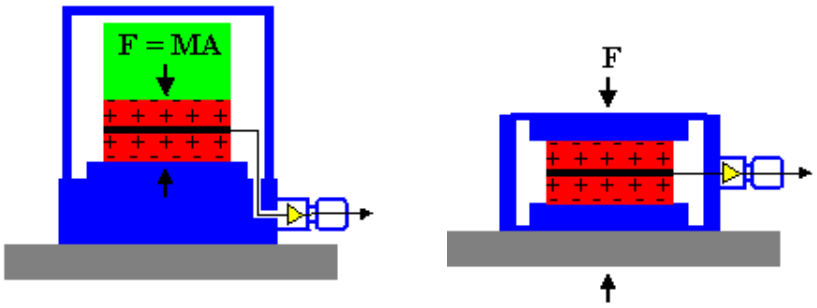
2.3.3 Data Acquisition

As mentioned above, a prerequisite for the acquisition of data from a test structure is a decision on how to support and excite the structure. Once this has been done, transducers may be used to measure the required parameters, where after an analyser may be used to record and process the measured data.

A transducer can be anything that converts energy from one form into another, generally mechanical energy into electrical energy (Meirovitch, 2001). For the purpose of this specific project, the chosen transducers were accelerometers and a load cell which are both piezoelectric, see Fig. 2.4.

Accelerometers were used to measure the acceleration of the structure at its point of attachment. According to *PCB* (Tech PCB [Online]), piezoelectric accelerometers use the piezoelectric effect of quartz or ceramic crystals to produce an electrical output which is proportional to the acceleration applied. *PCB* (Tech PCB [Online]) also states that this is effected by a generation or accumulation of charged particles on the crystal, this charge being proportional to the applied force on the crystal. A known mass is built into the accelerometer so that, by using Newton's second law ($F = ma$), the acceleration can be measured.

Once an electrical output has been produced, the built-in microelectronics in the chosen accelerometers produce a low impedance voltage signal that is compatible with most readout equipment (*PCB* Tech PCB [Online]).



(a) Piezoelectric accelerometer (b) Piezoelectric load Cell

Figure 2.4: Piezoelectric transducers

A load cell will be used to measure the input force from the exciter. A piezoelectric load cell works on the same principle as an accelerometer, except for measuring the force instead of the acceleration. Hence, there is no need for a known mass to be built into the load cell.

Inman (2001) states that much of the analysis of modal testing is done in the frequency domain. As explained by Inman (2001), the analyser receives the signal from the measuring equipment as an analog time-domain signal which is converted into digital frequency-domain information. This information is then compatible with digital computing. These signals are converted via Fourier transforms.

The analyser represents the obtained data as a frequency response function (FRF). A FRF expresses a structure's response to an applied force as a function of frequency (Irvine, 2000). Thus FRFs are transfer functions expressed in the frequency domain, see Eq. 2.1 below.

$$H(\omega) = \frac{X(\omega)}{F(\omega)} \tag{2.1}$$

$H(\omega)$ is the transfer function as a function of the angular frequency ω. $F(\omega)$ and $X(\omega)$ are the input and response functions respectively. Each function is complex which can also be represented in terms of magnitude and phase. According to Inman (2001) many different transfer functions are used in vibration measurement depending on whether displacement, velocity or acceleration is measured. The table below illustrates various transfer functions.

Table 2.1: Transfer functions used in vibration measurement (Inman, 2001)

Response Measurement	Transfer Function	Inverse Transfer Function
Acceleration	Accelerance	Apparent Mass
Velocity	Mobility	Impedance
Displacement	Receptance	Dynamic Stiffness

The three transfer functions given in Table 2.1 are all related to each other by algebraic equations (Irvine, 2000). Any of the functions can be calculated from another.

According to Inman (2001), the natural frequencies, damping ratios and modal amplitudes of a structure can be calculated from each peak of the measured FRF.

Coherence plots are used in conjunction with the FRF plots. A coherence function is a measure of the linear dependence between two signals as a func-

tion of frequency (Formenti, 1999). The coherence function is defined by the following equation

$$\gamma^2 = \frac{|S_{xf}(\omega)|^2}{S_{xx}(\omega)S_{xf}(\omega)} \tag{2.2}$$

Where $S_{xf}(\omega)$ is the cross power spectrum between the response and the input signals. $S_{xx}(\omega)$ and $S_{ff}(\omega)$ are the power spectrums of the response and input signals respectively.

According to Inman (2001) coherence is a measurement of noise in a signal. If the measurement is zero then the signal is pure noise. If the coherence value is one then the signal x and f are free of noise. In practise coherence is plotted against frequency and gives an indication of how accurate the measurement process is over a given range of frequencies. According to Inman (2001), values of γ^2=1 should occur at frequencies far from a structure's resonant frequencies. Near to the resonant frequencies the signals are large and magnify the noise. Inman (2001) suggests for practical purposes that if data has a coherence value of less than 0.75 it should be discarded.

Coherence plots are used to ensure that the peaks attained from the FRF plots are resonance frequencies. Inman (2001) states that the coherence function at the suggested natural frequency will be less than unity as the signals are increased at resonance and will in turn increase the noise in the signal.

LMS's Test.Lab will be used for data acquisition and as the analyser in this project. This program enables the user to access an array of outputs, including frequency response and coherence functions. It also allows for the assembling of models in order to see how the structures deform. The program has many other functions, of which the most relevant ones used during testing will be discussed in Chapter 5.

2.4 Data Comparison

A direct comparison between the measured and predicted data needs to be made in order to determine the differences and similarities between the two sets of data. The aforementioned needs to be known before the matching of the data can proceed. Ewins (1984) suggests that as many comparisons as possible should take place, one of the suggested comparisons being that between the three different models associated with dynamic data, i.e. the spatial, modal

and response models.

Comparing the spatial models is a matter of comparing the models' physical properties. Without physical tests, which may be destructive, finding the correct values of the physical model may be difficult. Therefore, the aforementioned values are assumed to be correct unless a correlation between the other two models cannot be found. In case of the latter, alternative non-physical methods can be applied, including the satisfying of the orthogonality requirements of measured modes, the adjustment of measured mode shapes, the adjustment of theoretical stiffness matrices, etc.

The comparison of response properties requires a comparison of the FRF gained from the measured data with that gained from the predicted data. Dascotte and Strobbe (1998) state that, theoretically, if noise-free measurements are available, the FRF at each frequency represent an equation of motion that accurately governs the system's dynamic behaviour. These FRF plots allow the engineer to compare the points of resonance and the amplitudes at these points. Ewins (1984) suggests that possible discrepancies in this data may be attributed to localised errors (loss of stiffness at joints, etc.) or to incorrect physical properties such as the elastic modulus or the material density.

However, obtaining the FRF for the numerical model may be difficult as all or at least a large portion of the modes needs to be included in the analysis. This requirement tends to cause some difficulty for certain analysis packages. Analysis packages can, however, easily calculate the modal properties of a specified number of modes without first having to calculate the FRF. A direct comparison between natural frequencies and mode shapes can be attained from such modal properties.

Another aspect to consider is that modal displacements or deformation information calculated by FE analyses can be misleading (FEAdomain, 2007). These displacements are calculated from eigen vectors which are arbitrarily scaled, so this does not necessarily provide good insight into the response. FEAdomain (2007) suggests that the modal effective mass can be employed to improve the understanding of the structure's response. The modal effective mass represents the fraction of a system's mass that is participating in its associated deformation mode. Consequently the effective modal mass provides a method for determining the significance of a deformation mode. Modes with relatively high effective modal masses can readily be excited by base excitation, while modes with low effective modal masses cannot readily be

excited by base excitation (Irvine, 2009). FEAdomain (2007) suggests that a mode's translational effective mass should be greater than 2 % before that mode is identified as a target mode. However this criterion can vary with application and proper engineering judgement should be implemented when applying this criterion.

The comparison of the mode shapes may be done by employing either graphic or numerical methods. One of the most useful ways of comparing mode shapes is by using the modal assurance criteria (MAC). The aforementioned provides a measure of consistency (degree of linearity) between modal vectors (Allemang, 2003). The MAC is calculated using the equation below

$$\mathbf{MAC}(p,x) = \frac{|\sum_{j=1}^{n}(\boldsymbol{\phi}_x)_j(\boldsymbol{\phi}_p)_j^T|^2}{(\sum_{j=1}^{n}(\boldsymbol{\phi}_x)_j(\boldsymbol{\phi}_x)_j^T)(\sum_{j=1}^{n}(\boldsymbol{\phi}_p)_j(\boldsymbol{\phi}_p)_j^T)} \tag{2.3}$$

where $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_p$ are matrices which represent the experimentally-measured and theoretically-predicted mode shapes.

The result supplies a matrix of values, so enabling all the test modes to be compared to any one of the numerical modes. A value of zero represents no correlation between two modes. A value of one represents a perfect match between two modes.

2.5 Optimisation

2.5.1 Numerical Design Optimisation

According to work by Schmit and Miura (1976), numerical optimisation is, essentially, a case of mathematical programming which provides a very general framework for scarce resource allocation . The problem statement of numerical design optimisation is very closely related to the problem statement of traditional engineering problems. As a result of the aforementioned similarity, the design tasks to which numerical design optimisation can be applied are inexhaustible.

Traditional design approaches made use of graphs and charts to evaluate and compare different designs (Vanderplaats, 2007). Although the methods employed in these approaches were effective, the increase in the size and complexity of design problems necessitated the use of computers to evaluate different solutions. Computers furthermore enable a designer to evaluate more

designs relatively quickly. Each time a designer wants to change an aspect of a design, however, a new set of variables has to be entered into the computer and the analysis has to be rerun. The next step in improving the design approach would be to automate the process of finding new sets of variables and rerunning the analysis with these new variables. Numerical design optimisation allows for this automation.

The generality and versatility of numerical design optimisation render it an unique tool for design automation (Design Manual *GENESIS*, 2008). Not only does it quicken the evaluation of a large amount of diverse designs, it also aids in searching for the most ideal solution within the parameters set by the designer.

2.5.2 General Optimisation Formulation

Vanderplaats (2007) gives the most general form of an optimisation problem:

The goal is to find a set of design variables X_i, $i = 1,l$ contained in vector $\mathbf{X}$ that will

$$\text{Minimise} \qquad F(\mathbf{X}) \tag{2.4}$$

Subject to:

$$g_j(\mathbf{X}) \leq 0 \qquad j = 1, m \tag{2.5}$$

$$h_k(\mathbf{X}) = 0 \qquad k = 1, n \tag{2.6}$$

$$X_i^L \leq X_i \leq X_i^U \qquad i = 1, l \tag{2.7}$$

where

$$\mathbf{X} = X_1, X_2, ..., X_l \tag{2.8}$$

Equation 2.4 is the objective function. The optimiser tries to minimise this function by changing the design variables in Eq. 2.8. Inequality (Eq. 2.5) and equality constraints (Eq. 2.6) can be implemented which must be satisfied by the final set of design variables. Upper and lower bounds (Eq. 2.7) can be placed on the design variables which will limit the search area.

2.5.3 Iterative Optimisation Algorithm

An initial set of design variables, $\mathbf{X}^o$, is required by most optimisation algorithms (Vanderplaats, 2007). Using this set as a point of departure, a design is updated iteratively until an optimum is reached. The iterative process used by most gradient based optimisation algorithms of updating these design variables is illustrated by the following equation:

$$\mathbf{X}^q = \mathbf{X}^{q-1} + \alpha^* \mathbf{S}^q \tag{2.9}$$

Where q is the iteration number, $\mathbf{X}^{q-1}$ is the current design point and $\mathbf{X}^q$ is the updated design point. $\mathbf{S}$ is a vector search direction in the design space and α^* is a scalar quantity which defines the magnitude of the search vector, i.e. how far the optimiser must move in the direction of $\mathbf{S}$.

When the optimiser is calculating $\mathbf{S}$, its goal is to try to minimise the objective function without violating the constraints. If X^{q-1} does not give a feasible solution (i.e. violating one or more constraints), then the optimiser will first search for a feasible solution, even if this increases the objective function. Once the optimiser has found a feasible solution, it will try to minimise the objective function.

After calculating $\mathbf{S}$, the optimiser does a 'one-dimensional search' to find an α^* value that will improve the design as much as is possible. Methods and techniques for calculating $\mathbf{S}$ and α^* are supplied in Vanderplaats (2007).

2.5.4 Advantages of Optimisation

Vanderplaats (2007) identifies the following advantages of optimisation:

- A decrease in design time
- An organised logical design procedure
- The capability of handling a large spectrum of design variables and constraints that is difficult to visualise using traditional methods
- A certain degree of design improvement ('almost always')
- An increasing of one's chances to obtain an improved, non-traditional design due to the fact that optimisation is not influenced by instinct or engineering experience

- A minimal required amount of interaction between the engineer and the computer program

2.5.5 Limitations of Optimisation

Vanderplaats (2007) identifies the following limitations of optimisation:

- Not only does an increase in the number of design variables affect an increase in computational time, it also causes the methods to become numerically ill-conditioned
- Unlike an engineer, optimisation techniques cannot rely on any previous accumulated engineering experience or on instinct, and are, instead, restricted to the specific scope of application of the analysis program
- It is obligatory to check results with great precision and care in the case of the analysis program not being theoretically accurate, as optimisation will typically exploit analysis errors in an attempt at improving mathematical designs
- Discontinuous functions prove to be a problem for the majority of optimisation algorithms whilst severe nonlinear problems may either converge very gradually or fail to converge at all. This demands great care to be taken in the formulation of the automated design problem
- As the attainment of the globally optimum design by the optimisation algorithm can most often not be guaranteed, restarting the optimisation process from a variety of points may be advantageous in providing a reasonable degree of assurance of the attainment of the global optimum
- As automated design is not necessarily considered in the writing of many analysis programs, some of these programs' analysis routines may need to be reprogrammed to include an optimisation code.

2.5.6 Structural Design Optimisation

Structural design optimisation is a specialised usage of numerical design optimisation that has been adapted to cater specifically for structural design problems. This project focuses on the special case of structural design optimisation.

The methods that form the basis of most modern optimisation were developed roughly 30 years ago, but much of the research in structural design since 1975 has been devoted to creating methods that are efficient for structural design problems where the analysis is expensive. This has resulted in various approximation methods that allow a high degree of efficiency while maintaining the essential features of the original problem.

Structural design optimisation has been developed to automate the design process by removing the need to repeat unnecessary FE analysis. The latter type of optimisation makes use of an approximation of the original problem. This approximate problem is solved by the optimizer and reduces the overall cost of structural design, as it is no longer necessary to repeatedly call the FE analysis during the actual optimisation process.

GENESIS is a finite element package that has been developed to implement structural design optimisation. *Design Studio* is a graphic user interface that can be used to set up optimisation problems. These programs were developed by *Vanderplaats Research and Development, Inc* and are the programs that will be employed in this project to apply the optimisation techniques.

GENESIS has five different optimisation techniques, namely shape, sizing, topology, topometry and topography optimisation techniques. Although only shape, sizing and topometry techniques were used in this project, a brief summary of all 5 the techniques will be given below.

Shape Design Optimisation

Shape design optimisation, as the name suggests, involves changing the shape of a FE model. The shape change is achieved by shifting the grid locations of a FE model. *GENESIS* has two methods for applying shape optimisation, namely the natural basis vector method and the domain method (Design Manual *GENESIS*, 2008).

Basis vectors represent alternative grid locations or shapes for a FE model (Candan, 2000). Linear combinations of perturbation vectors are calculated in order to determine the change in grid locations (Leiva and Watson, 1999). These vectors are associated with design variables that control their magnitude.

The new node locations are calculated as follows:

$$\mathbf{X}_i = \mathbf{X}_i^0 + \sum_j \mathbf{DV}_i(\mathbf{XB}_i - \mathbf{X}_i^0) \tag{2.10}$$

$$\mathbf{Y}_i = \mathbf{Y}_i^0 + \sum_j \mathbf{DV}_i(\mathbf{YB}_i - \mathbf{Y}_i^0) \tag{2.11}$$

$$\mathbf{Z}_i = \mathbf{Z}_i^0 + \sum_j \mathbf{DV}_i(\mathbf{ZB}_i - \mathbf{Z}_i^0) \tag{2.12}$$

Where $\mathbf{X}_i$, $\mathbf{Y}_i$ and $\mathbf{Z}_i$ are the updated coordinates of a node i and $\mathbf{X}_{io}$, $\mathbf{Y}_{io}$ and $\mathbf{Z}_{io}$ are the initial coordinates. $\mathbf{XB}_{ij}$, $\mathbf{YB}_{ij}$ and $\mathbf{ZB}_{ij}$ are the components of the j^{th} basis vector associated with node i. $\mathbf{DV}_i$ is the value of the j^{th} design variable.

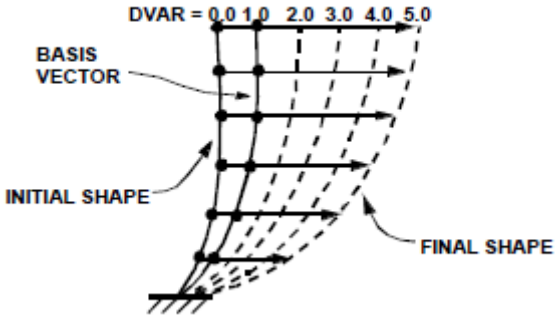


Figure 2.5: Different shape changes with a basis vector (Leiva and Watson, 1999)

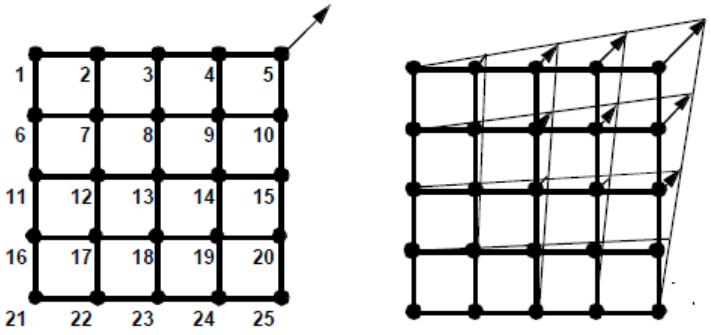
Figure 2.5 shows how the shape of a structure can be changed by a basis vector. As evident from the figure, the shape remains unchanged if the design variable's value is 0 (Candan, 2000). If the value is between 0 and 1, the new shape is interpolated between the original node and the locations of the nodes of the basis vectors (Candan, 2000). When the design variable has a value greater than 1 or less than 0, the new shape is extrapolated from the basis vector (Candan, 2000).

According to Leiva and Watson (1999), the advantages of the above mentioned method are that it is easy to use and quick to generate the input data. The user will not, however, have complete control over the interior grids' movement when using this computationally expensive method.

The second method possible in *GENESIS* is the domain method, which requires the addition of non-structural areas, termed "domains", to a FE model. Leiva and Watson (1999) describe the domains as representing finite elements that are normally large and contain many grids. The user applies perturbation vectors to the corners and/or the mid-sides of the domain in order to implement the desired shape change. All interior grids move along with those grids that have vectors applied to them. The aforementioned grids' movement is calculated using interpolation shape functions (Leiva and Watson, 1999).

Leiva and Watson (1999) claim the advantage of this method to be its ease of use and the designer's complete control over where the interior grid will move. This method is computationally cheap as there is no need to solve system equations. *GENESIS* has 10 different types of domains that can be used. For detail on these domains, refer to Design Manual *GENESIS* (2008).

The figure below shows an example of a quad domain, the domain most frequently used in this project.



(a) Input data

(b) Shape change for quad domain

Figure 2.6: Domain method (Leiva and Watson, 1999)

Sizing Design Optimisation

Leiva (2008) states that, in sizing optimisation, an element's cross-sectional dimensions are specified as design variables. The aforementioned allows the optimiser to change element properties by changing the element's cross-sectional dimensions (height, width, etc.). This is illustrated in the equations below:

$$A = A(W, H) = WH \tag{2.13}$$

$$I_x = I_x(W, H) = \frac{WH^3}{12} \tag{2.14}$$

W and H, the width and height of an element respectively, are entered into the optimiser as design variables. The optimiser changes these values, which in turn change element properties such as area (A) and inertia (I_x). This is advantageous as FE models calculate such properties. These properties can furthermore be easily designed by changing the element's cross-sectional dimensions.

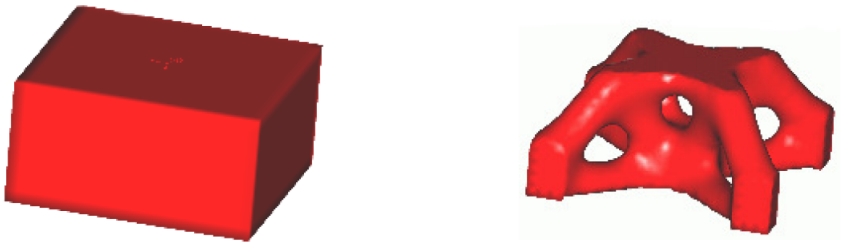
Sizing optimisation is applied to specified domains, whilst all the elements within each specific domain are assigned the same design variable.

Topometry Optimisation

Leiva (2008) states that topometry optimisation is a specialised sizing optimisation technique. In sizing optimisation, all elements are associated with a given property group and are thus designed identically. Topometry optimisation designs the elements on an element rather than property level, which allows for each element to be designed individually.

Topology Optimisation

Topology optimisation is used to identify which regions of the structure contribute to the overall stiffness of the structure and/or the natural frequencies that have been identified (Design Manual *GENESIS*, 2008). This optimisation technique is primarily used to create a stiff and light structure. Unlike shape and size optimisation, an initial design is not required for topology optimisation. This type of optimisation creates design variables associated with the Young's modulus and density of each element in the chosen region. The values for the design variables range from 0 to 1, where 1 indicates an element with normal mass and stiffness, and 0 indicates an element that has no mass or stiffness. The figure below gives an example of a result of topology optimisation:



(a) Initial design (b) Final design

Figure 2.7: Topology optimisation (Design Manual *GENESIS*, 2008)

Topography Optimisation

Topography optimisation is a technique normally applied to shell or composite elements that allows for the improvement of a structure's curvature (Leiva, 2007). It is considered a special kind of shape optimisation as it also uses perturbation vectors to shift grid locations. Leiva (2007) states that an important application of topography optimisation is that of bead pattern optimisation which is used to increase the stiffness of shell structures.

2.5.7 Data Matching

Optimisation allows for the automation of data matching as well as the simultaneous matching of all dynamic characteristics. Friswell and Mottershead (1993) provide an overview of the many methods that are available for updating FE models, including FRF-based methods. The aforementioned hold advantages over using modal parameters, as FRFs are free from curve-fitting errors and provide damping characteristics on an entire frequency spectrum directly (Dascotte and Strobbe, 1998). Although these methods are effective, they cannot match all dynamic characteristics simultaneously. Dascotte and Strobbe (1998) state that their proposed FRF-based updating method allows for the simultaneous updating of mass, stiffness and damping, but that this rarely proves successful. They suggest using a two step process instead. Firstly, the mass and stiffness properties are updated, followed by the damping properties.

GENESIS has two methods for matching data, namely the Least Squares Method and the Beta Method.

The Least Squares method minimises the sum of the squared, normalised,

differences between experimental and numerical data.

$$F = \sum_{i=1}^{NR} [M_i(R_i - T_i)]^2 \tag{2.15}$$

where T_i is the target value wanted, R_i is the calculated response, NR is the number of responses to be matched and M_i is a multiplier calculated using the following expression:

$$M_i = \frac{W_i}{\text{Max}(|T_i|, RMATCH)} \quad \text{if } (W_i > 0.0)$$

$$M_i = |W_i| \quad \text{if } (W_i < 0.0) \tag{2.16}$$

$$M_i = \frac{1.0}{\text{Max}(|T_i|, RMATCH)} \quad \text{if } (W_i = \text{blank})$$

where W_i is a weighting factor and $RMATCH$ is the minimum response/target normalising parameter, its default value is 0.01 and it must be larger than 0.

The Beta method minimizes the maximum difference between the experimental and numerical data.

$$\text{Minimse} \qquad \beta \tag{2.17}$$

Subject to

$$-\beta \leq M_i(R_i - T_i) \leq \beta \tag{2.18}$$

The Least Squares method was used in this project as all the modes needed to be matched. The goal was not to minimise a single error. *GENESIS*'s mode tracking algorithms allow for the tracking of modes regardless of frequency, which aids the simultaneous matching of all required modes.

Chapter 3

Scale Model of the Antenna Structure

A physical model of the antenna structure was required in order for vibration tests to be carried out. This model is a simplified, scaled model of the KAT-7 antenna structure. Tests conducted on the physical model provided the data that had to be matched with the FE model of the physical structure.

Detailed as well as assembly drawings of the physical model are provided in Appendix A.

3.1 The KAT-7 Antenna Structure

The KAT-7 project, constituting the first phase of the MeerKAT project, will involve the construction of seven 12 m radio telescopes. It will allow the South African SKA team to check and develop the design of the radio telescopes.

As illustrated in Fig. 3.1, each KAT-7 antenna structure consists of a round steel support structure, a steel yoke, a counterbalance structure, a dish as well as support legs for the feed (Bauermeister and More, 2008). The support structure is a rolled, welded steel structure bolted onto a solid foundation. The yoke, a second rolled, welded structure is located on top of the azimuth ring gear at the top of the support structure. A laser cut, bent steel backing structure fitted with a counterweight is located on the rear of the dish, a composite honeycomb sandwich structure. Two bearings on the yoke provide the dish with a pitching capability. Standard tubular beams make up the support legs for the feed.

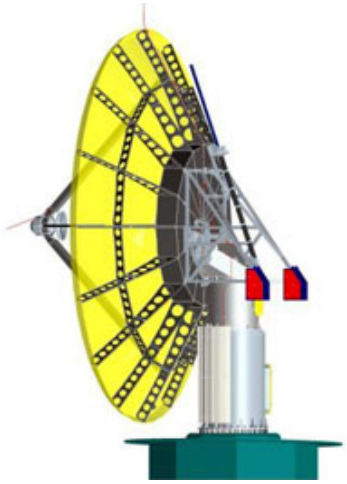


Figure 3.1: KAT-7 antenna structure (KAT-7 Update [Online])

3.2 Design of the Physical Model

The physical model was designed as a scaled, simplified model of the KAT-7 antenna structure. The dish diameter is 1.2 m, making the physical model a 1:10 scaled model of the KAT-7 antenna structure. An analysis of the most relevant aspects of the latter that needed to be represented on the physical model revealed the dish with support arms and feed to be of the greatest importance for vibration testing. These parts of the structure all contribute to the manner in which a radio telescope collects incoming signals, the base simply providing support to the previously named parts.

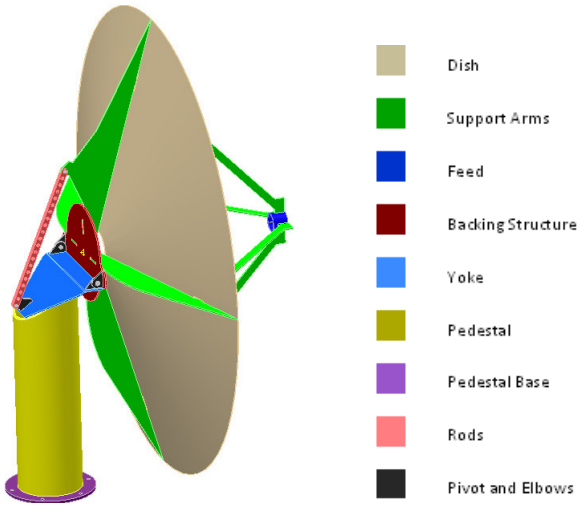


Figure 3.2: Physical model

The physical model was designed to consist of two separate parts, namely the support structure and the dish structure. The support structure consists of a pedestal, pedestal base, a yoke in two parts, 4 yoke elbows and a pivot. The dish structure consists of a backing structure, 4 support arms, a dish and a feed piece. Two adjusting rods secure the dish in position. The pivot and the rods allow for adjustments in the dish's angle of elevation. This design was based on a model provided by SKA.

All parts are welded together, except for the connection between the support and the dish structure. The aforementioned two parts are bolted together through the yoke elbows, whilst the rods are bolted to the pivot as well as the top support.

Figure 3.2 shows the designed model with a each part represented in a different colour.

All parts were made from mild steel with the following properties:

Table 3.1: Material properties

Property	Symbol	Quantity
Young's Moduus	E	210 GPa
Poisson's Ratio	v	0.33
Density	ρ	7859 kg/m^3

3.3 Manufacture of the Physical Model

All the parts of the physical structure were laser cut, except for the cylindrical parts of the structure, i.e. the feed and the pedestal. The dish was laser cut as a flat piece of sheet metal and then rolled and welded together. The cylindrical parts were provided by the workshop in the Mechanical Engineering Department of Stellenbosch University where the assembling of the structure was also carried out. Figure 3.3 shows a side and rear view of the manufactured model.

Figure 3.3: Manufactured model

Chapter 4

Numerical Model (FE Model)

This chapter will provide a discussion of the FE models of the KAT-7 antenna structure and that of the simplified model. A FE model is used to analyse and gain an understanding of a structure's response to various factors. Of interest to this project is the dynamic response of the antenna to environmental conditions. The more accurate the FE model, the better the understanding of the structure will be. Optimisation can assist the engineer in improving the FE model.

4.1 KAT-7

A FE model of the KAT-7 was generated by the SKA project team in order to gain an understanding of the response of the antenna to various environmental conditions. This model is shown in Fig. 4.1 and Fig. 4.2. Bauermeister and More (2008) states that the FE model of the KAT-7 is modelled as follows: "the support structure and the yoke, being steel plate structures, are modelled with shell elements. The counterbalance structure is modelled with beam elements, with the counterbalances mo-

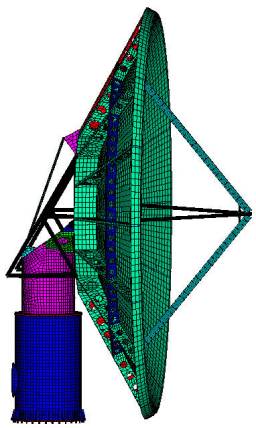


Figure 4.1: KAT-7's FE model - side view (Bauermeister and More, 2008)

delled as mass elements. The antenna dish is modelled as a sandwich, employing layered shell elements. The feed legs are modelled as beam elements, and the feed itself is modelled as a 1.5 m diameter disc with a mass of 100 kg."

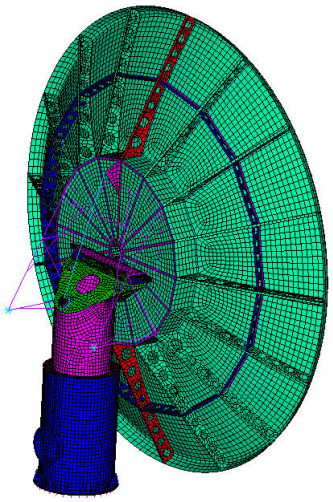


Figure 4.2: KAT-7's FE model - rear view (Bauermeister and More, 2008)

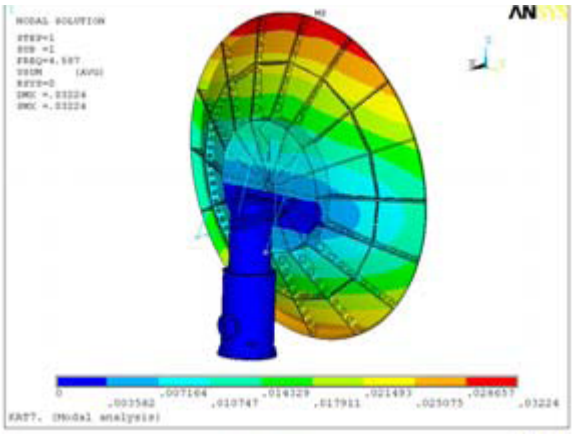
Bauermeister and More (2008) performed modal analyses on the KAT-7 FE model in 0^o and 90^o elevation positions. The specification on the lowest natural frequency for the antenna structure was 3 Hz. The results from the modal analyses are given in the table and the figures that follow:

Table 4.1 provides the frequency values for each mode at the two different elevation positions.

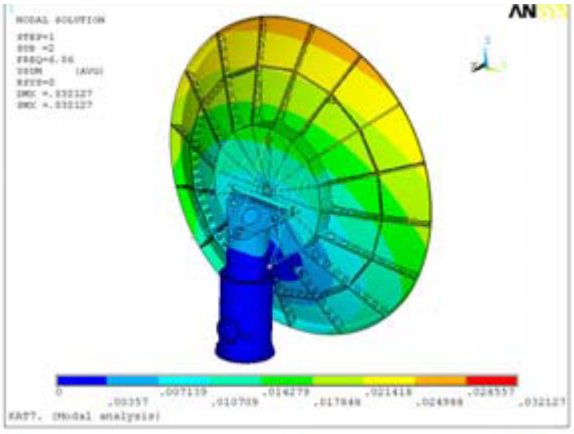
Table 4.1: Modal analysis of KAT-7

Mode no.	0° elevation	90° elevation
1	4.59 Hz	4.51 Hz
2	6.06 Hz	5.57 Hz
3	7.06 Hz	7.52 Hz
4	7.43 Hz	8.00 Hz
5	7.91 Hz	10.10 Hz

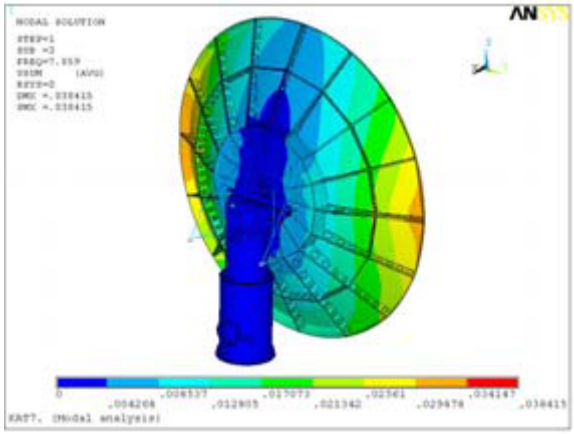
Figure 4.3 illustrates the mode shapes for the first five natural frequencies at an elevation of 0^o (Bauermeister and More, 2008).



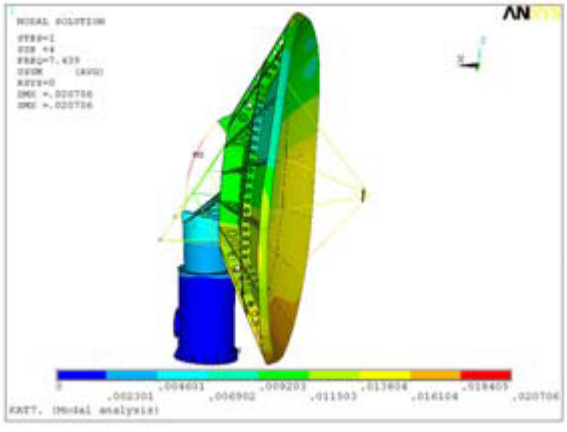
(a) Mode 1



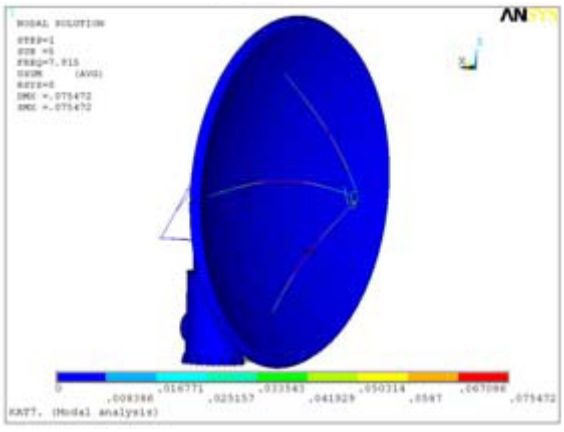
(b) Mode 2



(c) Mode 3



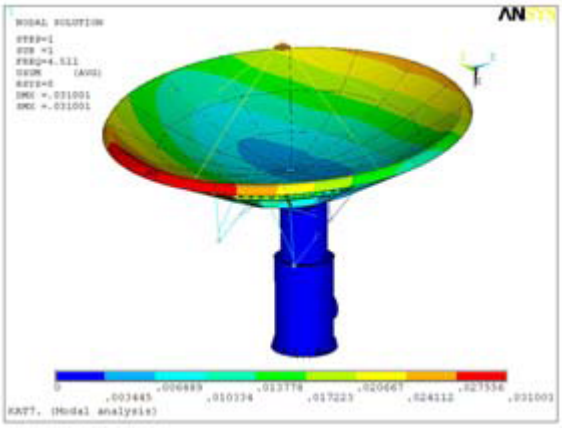
(d) Mode 4



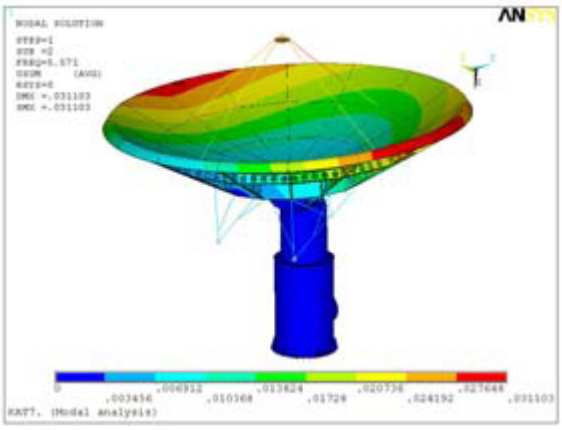
(e) Mode 5

Figure 4.3: First five natural modes of the KAT-7's FE model at 0^o elevation

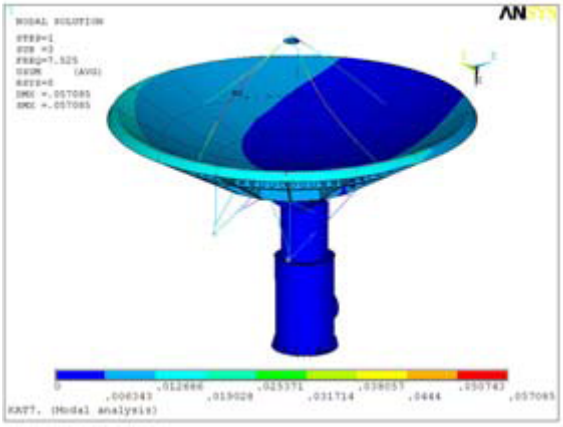
Figure 4.4 illustrates the mode shapes for the first five natural frequencies at an elevation of 90^o(Bauermeister and More, 2008).



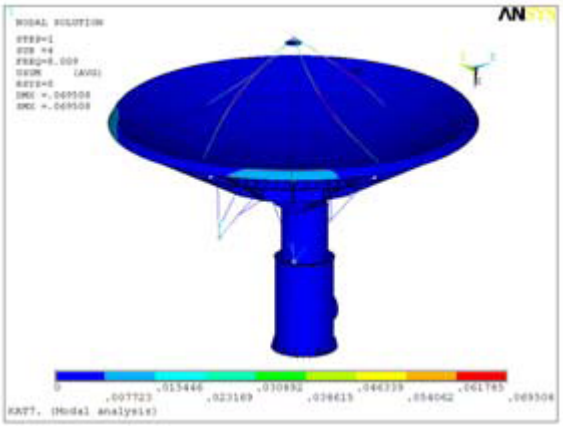
(a) Mode 1



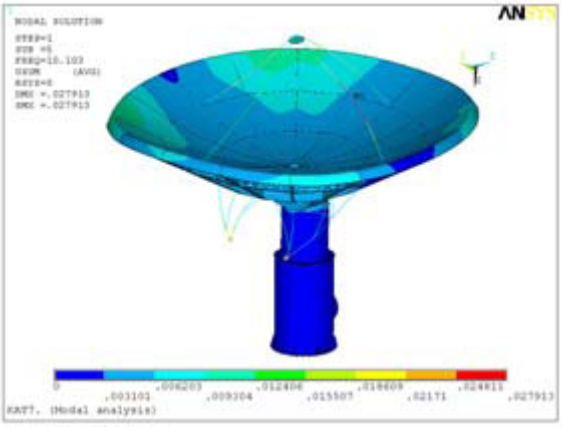
(b) Mode 2



(c) Mode 3



(d) Mode 4



(e) Mode 5

Figure 4.4: First five natural modes of the KAT-7's FE model at 90° elevation

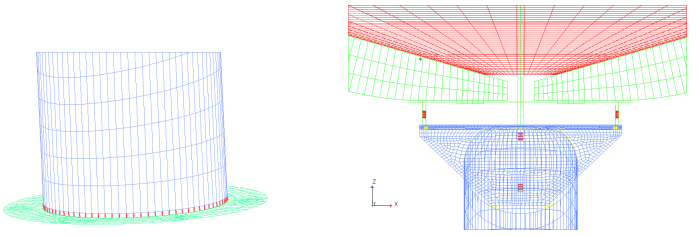
4.2 Simplified Model

4.2.1 Generation

The FE model of the simplified physical model was created using a combination of *Autodesk Inventor* and *MSC.Patran*. The geometry was created in *Inventor* and then saved as a STEP (Standard for the Exchange of Product Data) file. The latter was then imported into *MSC.Patran* in which the geometry was converted into appropriate surfaces and meshed. The FE model is illustrated below.

Figure 4.5: FE model of the simplified model

This FE model consists of 10 871 shell elements and 285 rigid body elements (RBE2s). The rigid body elements were used wherever two separate parts were welded together, e.g. where the pedestal was welded to the pedestal base, or where parts were bolted together, e.g. where the dish structure was bolted onto the support structure. The two previously mentioned examples are illustarted in Fig 4.6.



(a) RBE2 elements (red) connecting the pedestal (blue) to the pedestal base (green)

(b) RBE2 elements (red) representing bolted connection between support (blue) and the dish structure (green)

Figure 4.6: Rigid body elements

Literature suggests that the mass of the accelerometers should be added to the FE model as they may affect the model's dynamic characteristics. The masses of the accelerometers were added to the FE model using CONM2 (Concentrated Mass Element Connection), however this had a negligible influence so it was decided to excluded these elements. A BDF (Bulk Data File) file was exported from *Patran* and then changed to a DAT (Data) file so that it can be imported into *GENESIS*. Normal modes analysis was performed by *GENESIS*. Five modes were extracted as this is the number of modes analysed on the KAT-7.

4.2.2 Analysis

Before optimisation techniques could be applied, the FE model's output had to be compatible to that of the experimental data. The data to be compared were the frequencies and mode shapes. The frequencies were easy to compare as both the FE model and the vibration tests supply this data directly. The mode shapes, however, had to be derived from the displacements of the measured points used during the vibration tests. The mode shapes derived during these tests are based on the data received from 14 specific measurement points. In order to compare the FE model's mode shapes to that of the physical model, the same 14 points had to be located in the FE model. The mode shapes were thereafter derived from only these 14 points.

The output of the FE model provides displacements in an x-y-z plane whilst the vibration tests only provide displacements perpendicular to the surface on which the specific accelerometer is located. The 3 vectors of the FE model output were used to calculate one vector in the same orientation as that measured during vibration testing. This was done by using Python scripts to read in the output from the FE analysis and then calculating a vector which would be equivalent to the directions measured during the vibration tests. The latter allowed for a direct comparison of the degrees of freedom measured and calculated from the respective models.

Once the 14 points from both models were available in a compatible format, the mode shapes were plotted and compared visually, whereafter the MAC values were calculated. Python scripts were used to perform both these functions as well as the plotting of the MAC values in a MAC colour graph.

The above 14 points were later employed during the optimisation process,

a discussion of which can be found in Chapter 7.

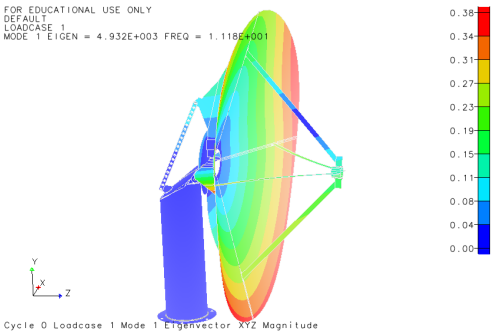
4.2.3 Results

A finite element analysis was performed on the simplified model in order to establish its dynamic characteristic. The analysis was performed on the model as a whole as well as on the dish and support structure as two separate parts.

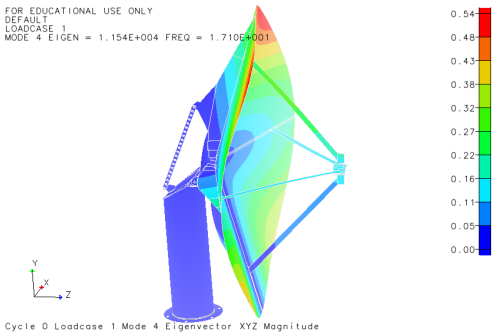
Complete Model

A FE analysis was performed on the whole structure at 0^o and 90^o elevation. The model was fixed at its base as it would be during operation.

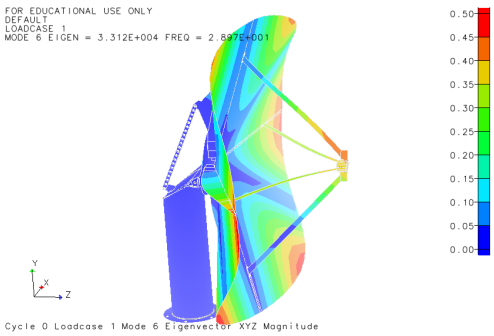
The following results were obtained from the modal analysis with the antenna in the 0^o elevation position:



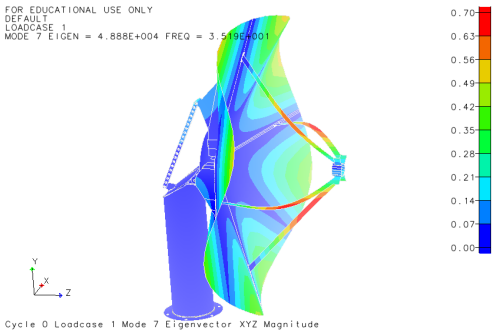
(a) Mode 1 at 11.18 Hz



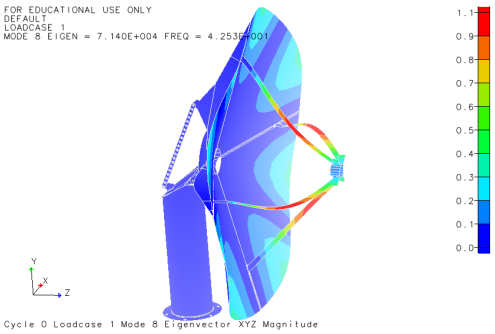
(b) Mode 2 at 17.10 Hz



(c) Mode 3 at 28.97 Hz



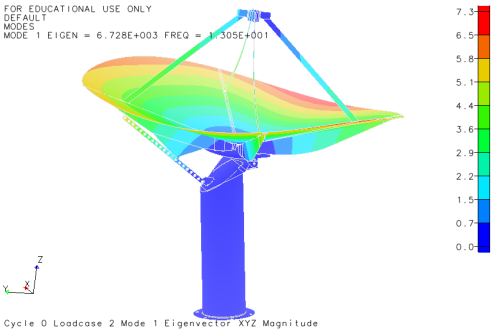
(d) Mode 4 at 35.19 Hz



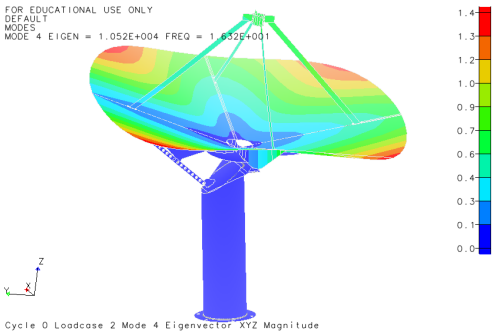
(e) Mode 5 at 42.53 Hz

Figure 4.7: First 5 modes of the whole simplified FE model at 0° elevation

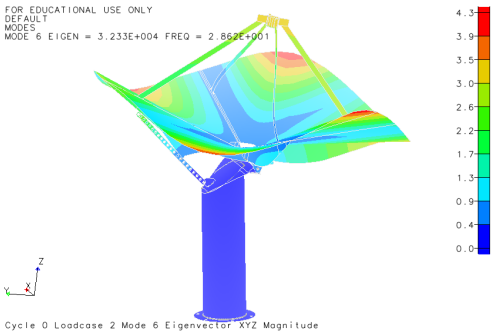
The following results were obtained from the modal analysis with the antenna in the 90° elevation position.



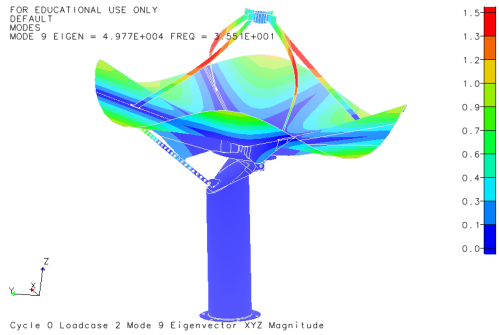
(a) Mode 1 at 13.05 Hz



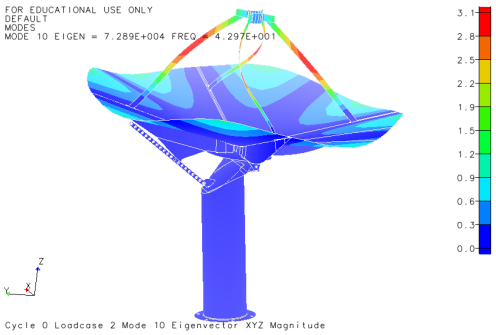
(b) Mode 2 at 16.32 Hz



(c) Mode 3 at 28.62 Hz



(d) Mode 4 at 35.51 Hz

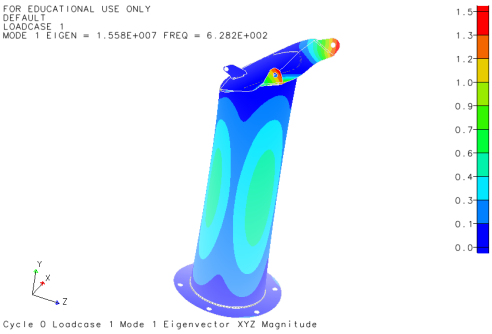


(e) Mode 5 at 42.97 Hz

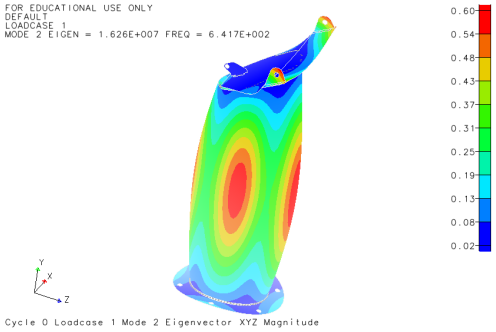
Figure 4.8: First 5 modes of the whole simplified FE model at 90° elevation

Support Structure

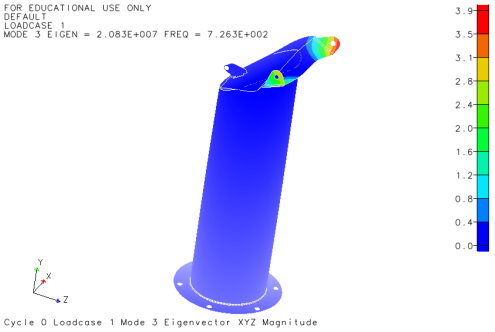
The modal analysis that was performed on the support structure was performed in two set-ups. The first involved no constraints on its degrees of freedom, i.e. it was conducted under 'free-free' support conditions. The second set-up contained constraints placed on the base of the pedestal as would be the case in operation. The following FE results were obtained for the support structure:



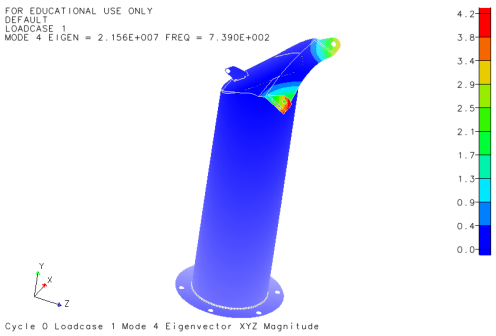
(a) Mode 1 at 628.20 Hz



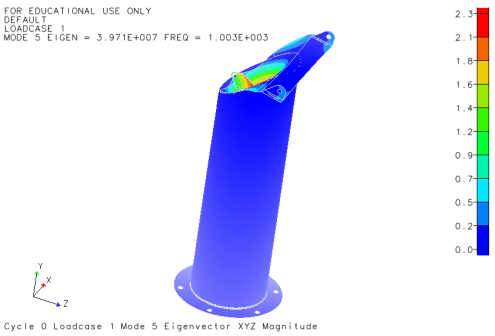
(b) Mode 2 at 641.70 Hz



(c) Mode 3 at 726.3 Hz

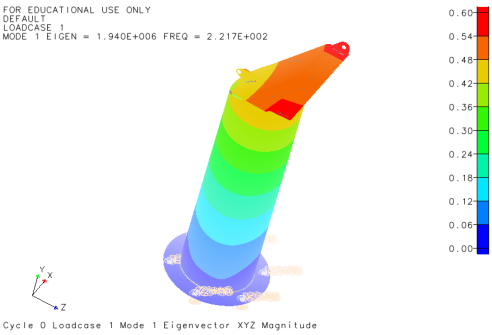


(d) Mode 4 at 739.00 Hz

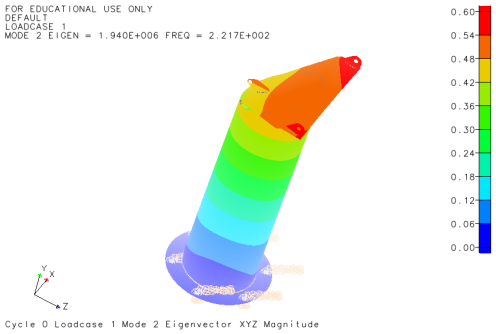


(e) Mode 5 at 1003.00 Hz

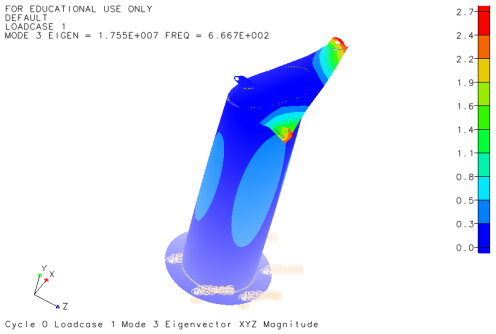
Figure 4.9: First 5 modes of the support structure ('free-free')



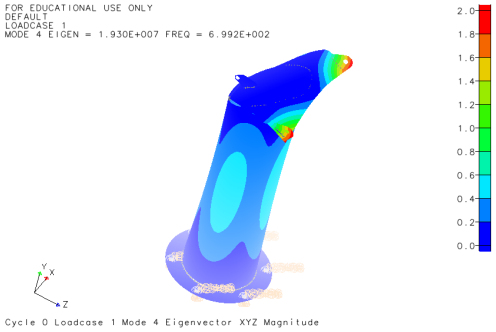
(a) Mode 1 at 221.70 Hz



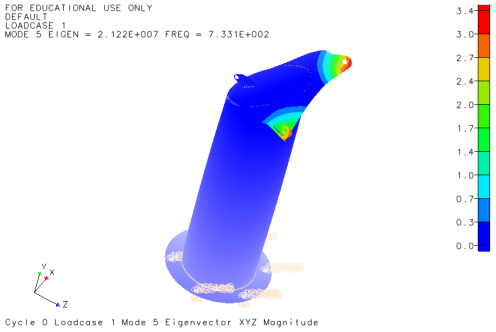
(b) Mode 2 at 221.70 Hz



(c) Mode 3 at 666.70 Hz



(d) Mode 4 at 699.20 Hz

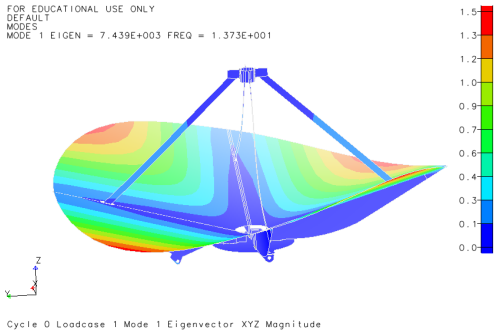


(e) Mode 5 at 733.10 Hz

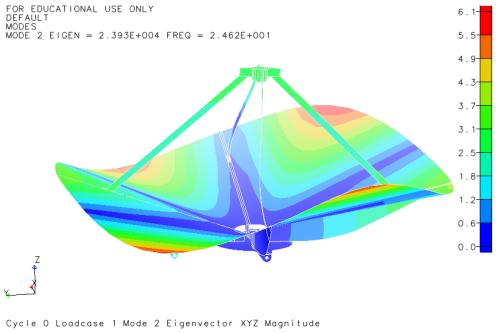
Figure 4.10: First 5 modes of the support structure (fixed base)

Dish Structure

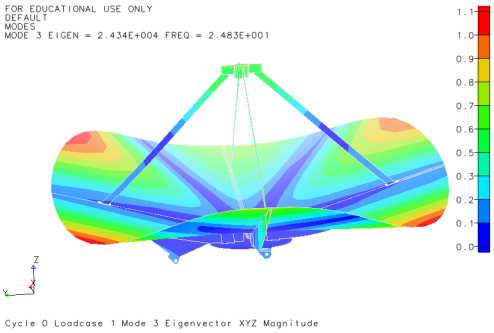
The modal analysis that was performed on the dish structure involved no constraints on its degrees of freedom, i.e. it was also conducted under 'free-free' support conditions. The following images were used to establish the best placement for the accelerometers in order to extract the mode shapes from the vibration tests:



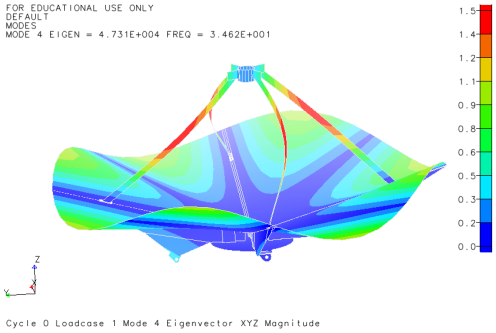
(a) Mode 1 at 13.73 Hz



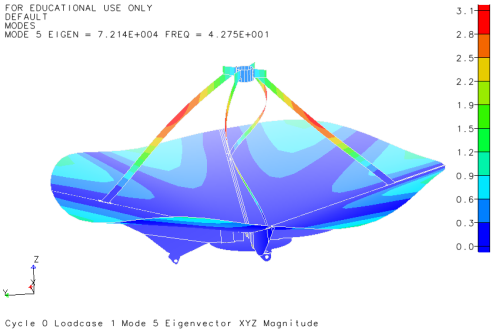
(b) Mode 2 at 24.62 Hz



(c) Mode 3 at 24.83 Hz



(d) Mode 4 at 34.62 Hz



(e) Mode 5 at 42.75 Hz

Figure 4.11: First 5 modes of the dish structure

Effective Modal Mass

The effective modal mass provides an indication of a deformation mode's significance and whether or not a specific mode can be excited by base excitation. Table 4.2 shows the effective modal mass for the dish seperate from it's pedestal, while Table 4.3 shows the effective modal mass for the structure as a whole.

Table 4.2: Effective modal mass for the dish

Mode	T1	T2	T3	R1	R2	R3
1	9 %	0 %	2 %	4 %	0 %	2 %
2	21 %	57 %	2 %	83 %	3 %	27 %
3	66 %	0 %	85 %	0 %	89 %	32 %
4	0 %	20 %	0 %	2 %	1 %	7 %
5	0 %	1 %	0 %	2 %	0 %	4 %

Table 4.3: Effective modal mass for the structure as a whole

Mode	T1	T2	T3	R1	R2	R3
1	1 %	0 %	0 %	0 %	1 %	30 %
2	0 %	0 %	0 %	0 %	0 %	3 %
3	0 %	29 %	1 %	8 %	0 %	0 %
4	15 %	0 %	0 %	0 %	49 %	7 %
5	6 %	0 %	0 %	0 %	0 %	4 %

4.2.4 Conclusion

In regards to the successful collection of incoming signals, the dish is of main concern as its shape variations and feed movement determine the manner in which the antenna receives such signals. As can be seen from the FE analysis results, the frequency ranges of the structure as a whole and that of the dish structure are similar, whilst the support structure's frequencies are far higher than either of the aforementioned. The highest frequency of the whole model is 42.97 Hz. In regards to the dish, the highest frequency is 42.75 Hz, whilst the lowest frequency of the support structure is 628.20 Hz in a 'free-free' support condition and 221.70 Hz with its base fixed. The far higher frequency range of the support structure indicates that the latter will have a very small influence on the first 5 modes of the dish.

Chapter 5

Vibration Testing

5.1 Introduction

This chapter will provide a discussion of the manner in which the vibration tests were carried out as well as the results thereof. The purpose of implementing tests on the physical structure was to obtain the structure's natural frequencies as well as its corresponding mode shapes. The tests were not completed on the structure as a whole, but on the dish structure alone. This decision was based on investigation done via FE analysis in Section 4.2.3. From the aforementioned investigation it could be seen that the support structure and the dish's frequencies differ immensely. The first five modes, being the only ones of interest to this project, all exhibited frequencies below 50 Hz. The support structure's first bending mode was at 628.20 Hz, whilst the dish's highest mode of interest had a frequency of 42.75 Hz. It can thus be deduced that the support structure will have a very small influence on the frequency range of interest. The dish structure is of more significance, as it is the main component responsible for the collection of incoming signals. Understanding the dish's dynamic characteristics will enable an analyst to secure the accuracy with which signals are collected under different environmental conditions.

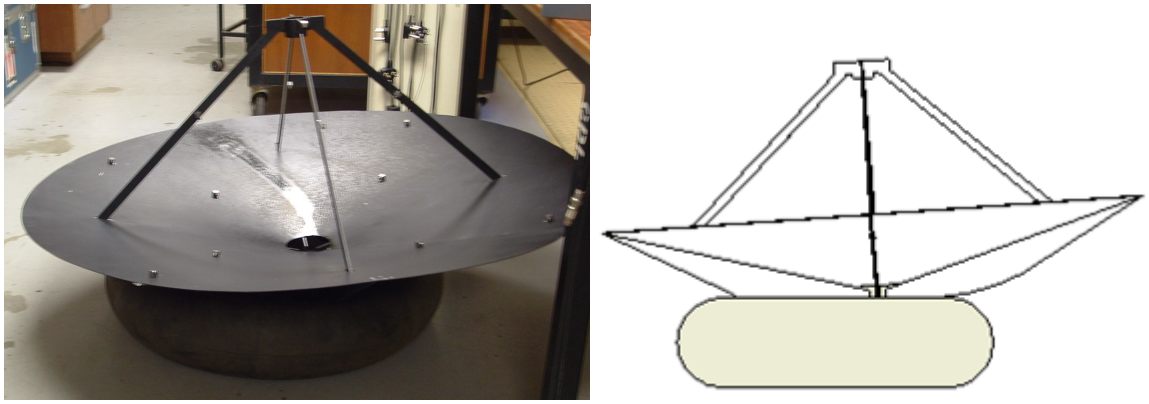
5.2 Experimental Set-up

5.2.1 Support

The dish was tested under 'free-free' support conditions. Such conditions are achieved by using a form of soft support, in this case an inflatable tube. Em-

ploying this specific form of soft support is common practice when conducting vibration tests as the tube's frequency range is much lower than that of the structure being tested. If the dish and support structure had been tested as a whole, its geometry would have complicated the achievement of 'free-free' support conditions.

The support set-up is shown in the figure below.



(a) Photo (b) Sketch

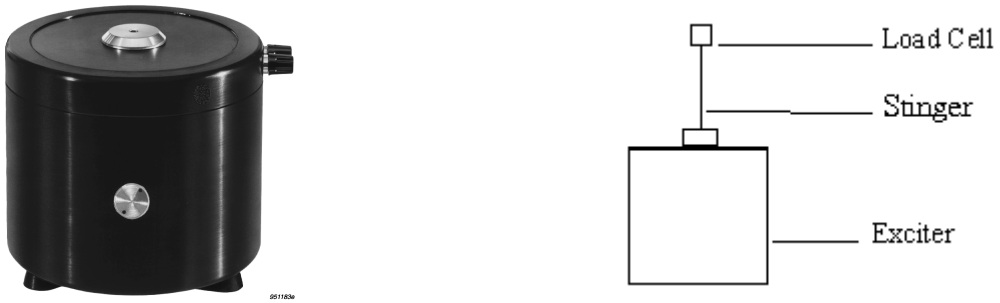
Figure 5.1: Support set-up for vibration testing

5.2.2 Equipment

The equipment set-up comprised of three sections, namely the excitation mechanism, the transduction system as well as an analyser.

Excitation Mechanism

An electromagnetic exciter was used to excite the dish's natural frequencies. The exciter was connected to the dish via a stinger and a load cell. Load cells are used in order to measure the input of the exciter. The load cell was attached to the rim of the dish half way between two support arms. The decision to place the load cell at this exact point was governed by the position of the accelerometers as they often get overload errors if placed too close to the excitation position. The exciter itself was fixed to the ground. The specifications for the exciter are provided in Appendix B.



(a) Photo (b) Sketch

Figure 5.2: Electromagnetic shaker

Transduction System

The transduction system consisted of 14 accelerometers and 1 load cell. The accelerometers were used in order to measure the frequencies and the mode shapes of the dish structure. The positioning of the accelerometers was determined after an evaluation of the FE model's mode shapes. From this evaluation the best positions for the recovery of these shapes from the physical structure could be estimated. This was done by selecting the grid points, for each mode, that were on the peaks of the displaced structure. From these grids, it was assessed as to which grids were able to best capture all the desired modes' shapes. The specifications of the accelerometers are provided in Appendix B.

The Analyser

The analyser system employed in this project was an *LMS* system. *LMS Test.Lab* was used to convey signals to the exciter, i.e. control the voltage output, and record the input from the accelerometers. Of the 15 channels used, one was linked to the load cell and 14 to the accelerometers. *LMS Test.Lab* recorded the data from the accelerometers and produced FRFs and coherence functions that were used to extract modal frequencies and mode shapes. A model of the test set-up was furthermore generated in *LMS Test.Lab* in order to supply a visual display of the manner in which different points deform relative to each other (see Fig. 5.3). These models were used for comparison with the FE model's modes.

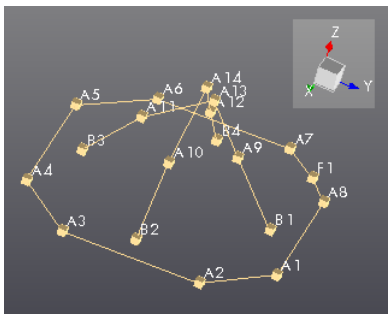


Figure 5.3: Model in analyser showing transducer positioning

A1 - A14 represent the accelerometer positions, whilst F1 represents the excitation point. B1 - B4 are the points where the support arms come through the dish. No measurements are taken at these points as they merely help to complete the model. For detail on the equipment specifications, see Appendix B.

5.2.3 Transducer Positioning

The specific transducer positions were selected in order to supply the mode shapes of all the desired modes. This set-up can be seen in Fig. 5.3 and in Fig. 5.4.

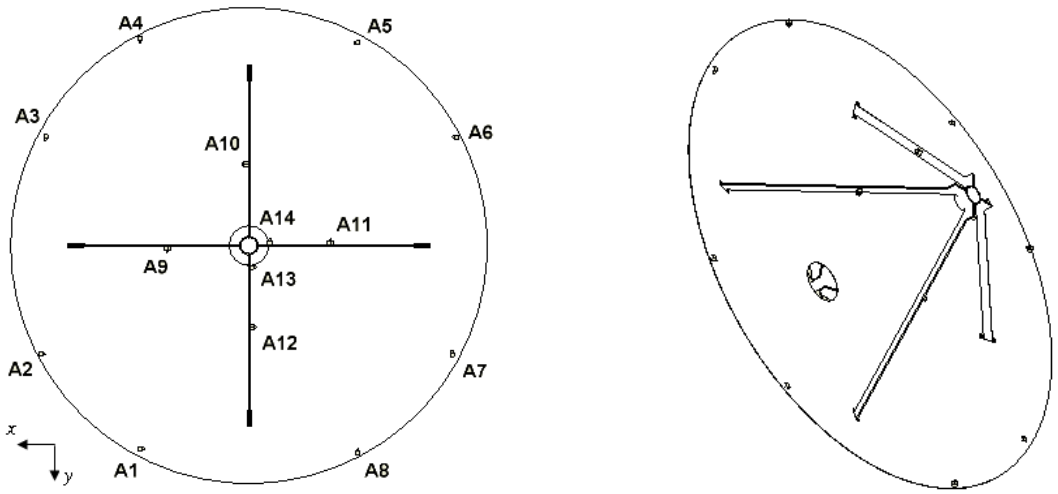


Figure 5.4: Accelerometer positioning

Eight accelerometers were placed at measuring points around the rim of the dish, two between each support arm, in order to track the dish's shape

variations. These accelerometers measured the displacement perpendicular to the dish's surface, a direction defined as the z plane for the purposes of these tests. The latter orientation results in the dish's surface being labelled the x-y plane with z = 0. The centre of the dish was defined as the origin.

One accelerometer was furthermore placed onto each of the four support arms. The feed itself had two accelerometers attached to it. These six accelerometers tracked the movement of the support arms and the feed piece in the x-y plane.

5.3 Experimental Procedure

5.3.1 Preparation

In preparation for the testing phase of this project, a FE model of the structure had to be completed as discussed in Chapter 4. This model provided insight into what should be expected from the tests and helped to determine the accelerometer and excitation positions. Once the measuring positions were finalised, the test set-up could begin. Firstly, the dish's surface was cleaned with acetone and the dish placed on the inflatable tube. Wax was next used to attach the accelerometers to the dish in their marked positions. The exciter was attached to the dish via the stinger and load cell, the latter of which was glued to the dish with an epoxy glue.

The accelerometers and the load cell were connected to the analyser. The load cell fed into channel 1, accelerometer 1 fed into channel 2, accelerometer 2 fed into channel 3, and so forth. The simplified model was next set up using the coordinates of the accelerometer positions (see Fig. 5.3). The coordinates were given in an x-y-z coordinate system. Lastly, the accelerometers' specifications were added. For the accelerometers' positions and specifications, refer to Appendix B.

5.3.2 Testing

Two separate measurements were taken, each with a bandwidth of 64 Hz. The second test was performed in order to check the repeatability of the test and to confirm the results of the first test.

5.3.3 Measured Results

The measured results consisted of FRF response functions and coherence plots for each accelerometer. At each resonance frequency, a summary was given of each accelerometer's rectangular (amplitude and phase) and polar (real and imaginary) units. The model that was set-up in *LMS Test.Lab* was also used as a visual aid in the analysis of the mode shapes.

FRF Plots

The signals received from the accelerometers were converted into the frequency domain by performing Fourier transforms. FRFs were used to obtain the natural frequencies, damping ratios and modal amplitudes. The following figure represents the FRF received from accelerometer 1.

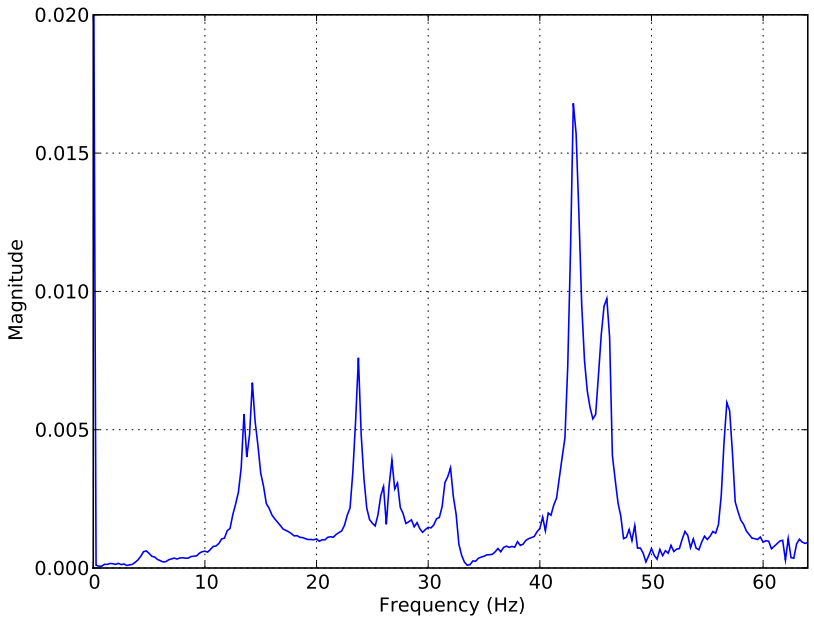


Figure 5.5: FRF for accelerometer 1

Before assessing the resonant frequencies for the deformation modes, the rigid body modes of the soft support needed to attained in order to confirm that they are low enough to ensure that it is 'soft' enough. The highest rigid body mode was at 2.25 Hz, which is 16 % of the first natural frequency. This satisfies the soft support requirement.

Coherence and Phase Plots

Not all the peaks on a FRF plot represent resonance frequency. Coherence and phase plots were used to ensure that these frequencies are in fact resonance points. Inman (2001) guidelines suggest that the coherence value at resonance should be lower than those away from resonance and the phase should be +/- 90 degrees. The figures below are examples of a coherence plot and a phase plot obtained from the tests.

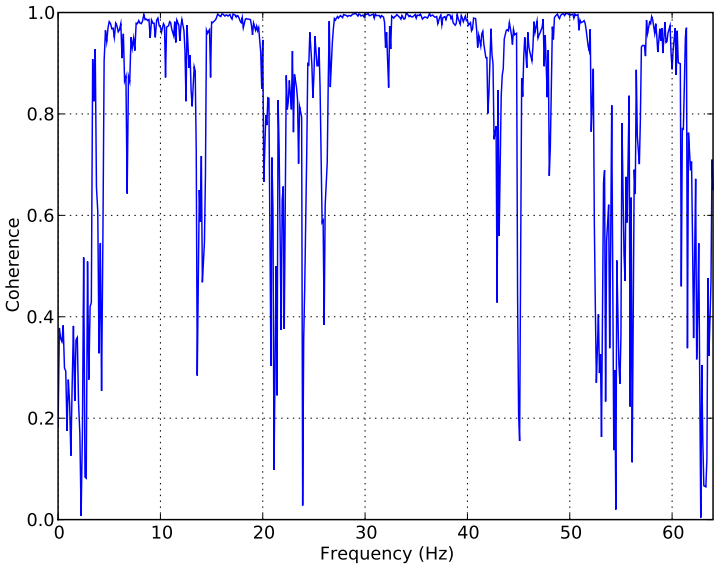


Figure 5.6: Coherence plot for accelerometer 1

Resonance Points Summary

Table 5.1 is an example of the output that can be extracted from *LMS Test.Lab*. It depicts the accelerometers and whether they took measurements in an x, y or z direction. It also shows the amplitude and phase measured by each accelerometer. As all accelerometers are at resonance, all phases are 90 or -90. The phase signs indicate whether the accelerometers moved in a positive or negative direction.

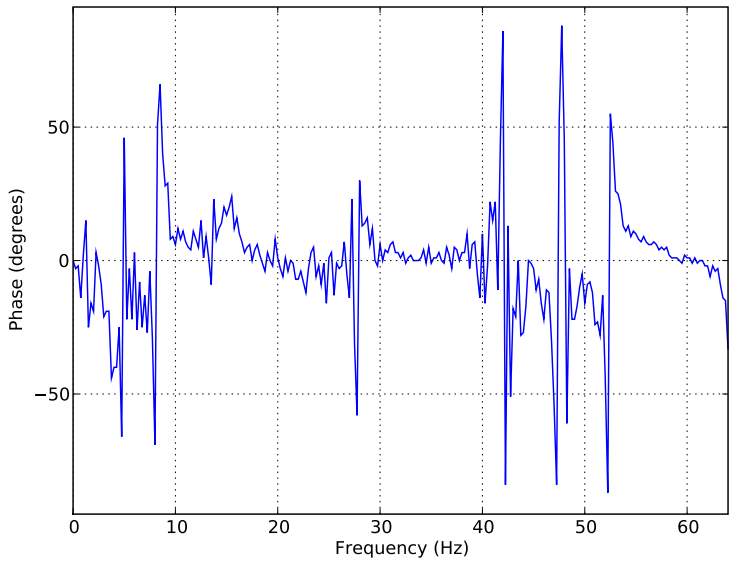


Figure 5.7: Phase plot for accelerometer 1

Table 5.1: Summary of resonance frequency at 13.93 Hz

Response	Magnitude	Phase	Real	Imaginary	Coherence
A1: +Z	7.15E-04	90	0	7.15E-04	0.64
A2: +Z	9.51E-04	90	0	9.51E-04	0.65
A3: +Z	8.44E-04	-90	0	-8.44E-04	0.68
A4: +Z	1.10E-03	-90	0	-1.10E-03	0.63
A5: +Z	8.50E-04	90	0	8.50E-04	0.68
A6: +Z	1.01E-03	90	0	1.01E-03	0.63
A7: +Z	8.60E-04	-90	0	-8.60E-04	0.69
A8: +Z	9.87E-04	-90	0	-9.87E-04	0.67
A9: +Z	1.72E-04	-90	0	-1.72E-04	0.42
A10: +Z	8.02E-05	-90	0	-8.02E-05	0.72
A11: +Z	1.14E-04	90	0	1.14E-04	0.58
A12: +Z	1.04E-04	90	0	1.04E-04	0.48
A13: +Z	2.84E-05	-90	0	-2.84E-05	0.25
A14: +Z	8.58E-06	-90	0	-8.58E-06	0.62

Test.Lab's Model

The model set-up in *Test.Lab* was employed as a visual aid for analysing the mode shapes of the physical test structure. The model illustrates the manner in which the different accelerometers move relative to each other in their respective directions. The model was viewed both as a deformed model (see

Fig. 5.8) and as an animation that shows an oscillating model. The figure below depicts the deformed model for the first mode.

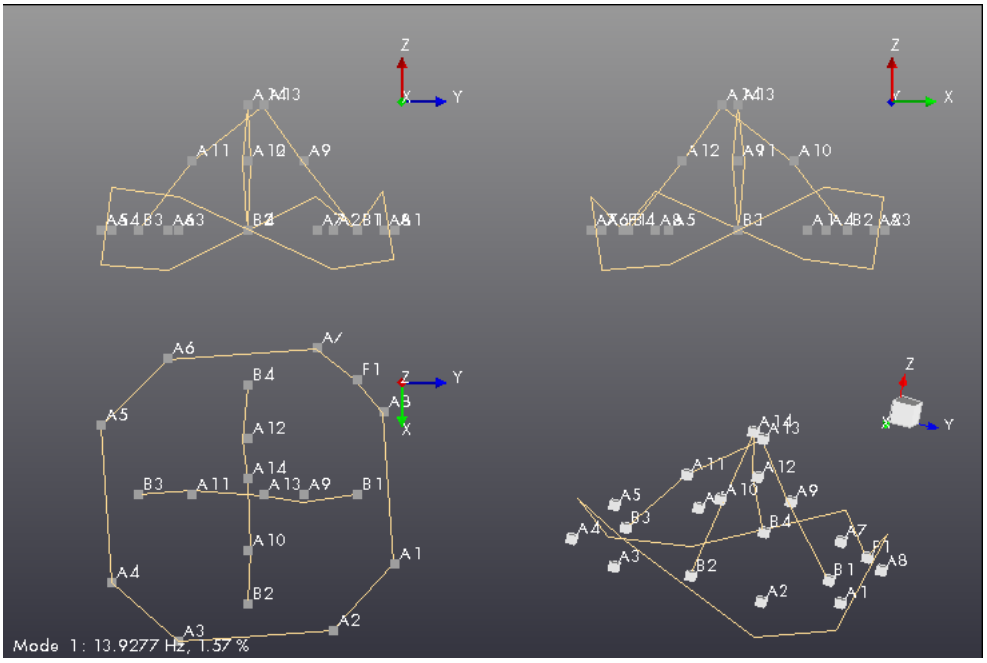


Figure 5.8: *LMS Test.Lab's* deformed model for the first mode

5.4 Processed Results

The results from the two tests were compared in order to check the repeatability of the test and to establish the accuracy of the obtained data. The comparison focused on the tests' frequencies and their mode shapes, the latter being compared using the modal assurance criterion. The results are supplied below in the form of a table of frequencies as well as a MAC plot.

Table 5.2: Frequencies comparison of the test runs

Mode	Run 1	Run 2	Error	MAC
1	13.93 Hz	13.82 Hz	0.79 %	0.996
2	23.61 Hz	23.64 Hz	0.13 %	0.953
3	26.15 Hz	26.21 Hz	0.23 %	0.998
4	32.06 Hz	32.05 Hz	0.03 %	0.858
5	42.13 Hz	42.14 Hz	0.02 %	0.998

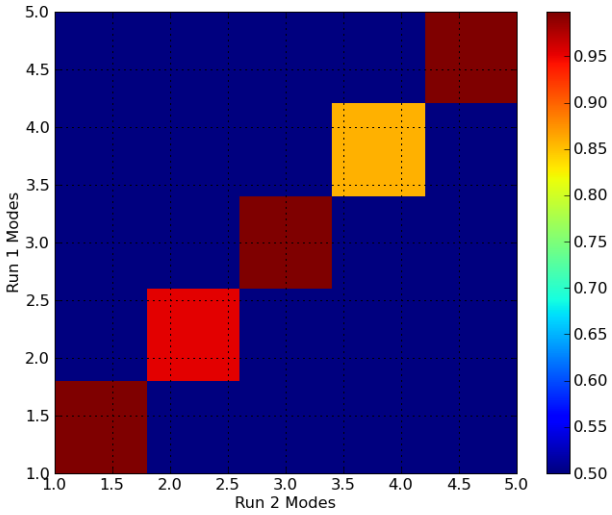


Figure 5.9: MAC of the test runs

5.5 Conclusion

The vibration tests were successful and the required data obtained. The two repetitions of the same test were similar enough to secure confidence in the accuracy of the data. The maximum error in the frequencies was 0.8 % and the worst match for the mode shapes had a MAC value of 86 %. The next step, discussed in the following section, was to compare the measured data with the numerical data and to determine in which ways the physical model and FE model differ. The measured data used for the comparison will be that of the first test. The tests were so closely related that taking the average would not have made a significant enough effect to make it necessary.

Chapter 6

Data Comparison

This chapter will provide a discussion of the manner in which the two models' data was compared as well as supply a summary of the initial comparison before optimisation was applied. Firstly, by simply contrasting the values thereof in Hz, the frequencies of the two models were compared. A comparison between the mode shapes proved more complex, however, thereby necessitating the use of a number of visual aids as well as MAC plots.

6.1 Frequency Comparison

The frequencies are easily attained from both the FE analysis and the vibration tests. The respective models' frequency values were compared directly as can be seen in Table 6.1.

Table 6.1: Frequency comparison between tests and initial FE model

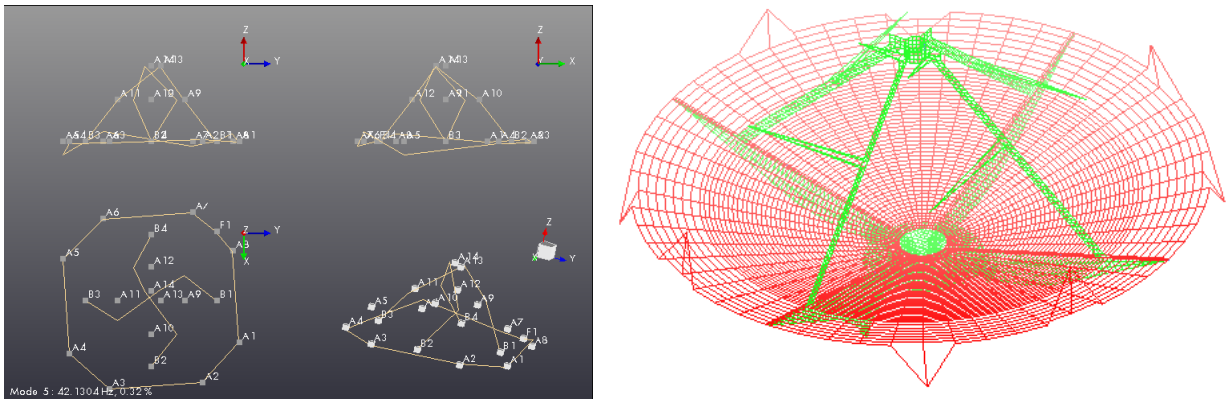
Mode	Initial FE model	Vibration Tests	Error
1	13.73 Hz	13.93 Hz	1.44 %
2	24.62 Hz	23.61 Hz	4.28 %
3	24.83 Hz	26.15 Hz	5.05 %
4	34.62 Hz	32.06 Hz	7.99 %
5	42.75 Hz	42.13 Hz	1.47 %

6.2 Mode Shape Comparison

The mode shapes were compared using visual aids and MAC calculations. The details of these two methods follow below.

6.2.1 Visual Aids

The first visual aid was a comparison between the deformation plots produced by the FE analysis and the deformation plots produced by *LMS Test.Lab*. *Test.Lab*'s model only shows the accelerometers' movement, so to make the comparison easier the FE analysis was set-up in a similar manner. This was achieved by requesting only the deformations of the grids that represent the accelerometer measurement positions. This is illustrated in Fig. 6.1b.

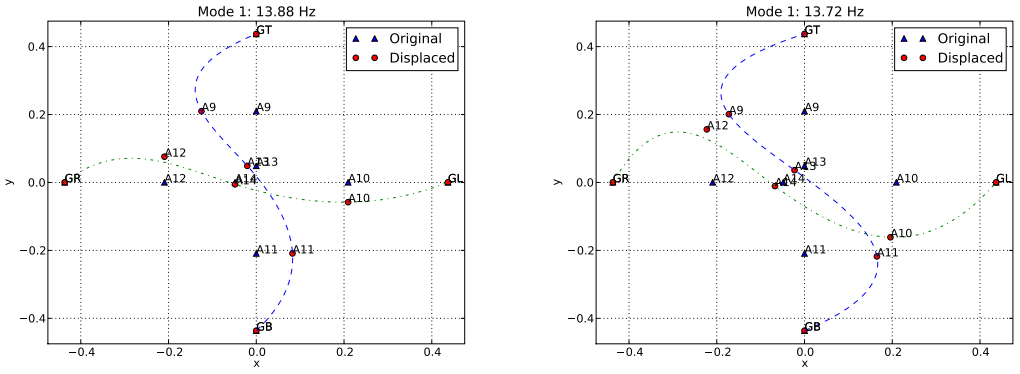


(a) *LMS* model mode 5 (b) FE model mode 5

Figure 6.1: Model comparison of mode 5

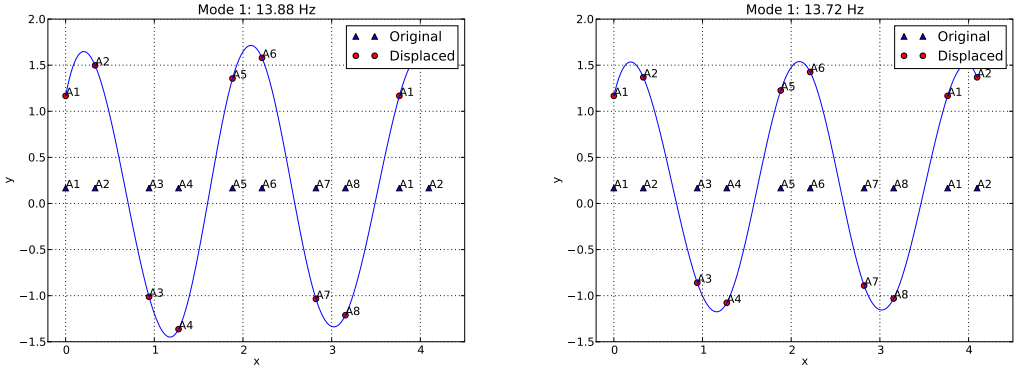
The second visual aid was the plotted mode shapes of the FE model and the physical tests. These plots were produced by using *Python* scripts that employ curve-fitting techniques to produce the mode shapes. The curve fitting technique used was *Scipy*'s class 'InterpolatedUnivariateSpline(x, y)'; this is a univariate spline s(x) of degree k on the interval [xb, xe] calculated from a given set of data points (x, y). The curve fit was based on points that represent the accelerometer-measured deformation. These plotted mode shapes are not the exact shapes that the physical structure will produce when the structure is exposed to external excitation, and were used purely for purposes of comparison.

The mode shapes were plotted in two separate 2-D planes to simplify the process of comparison. The first 2D-plane consisted of the measurement points on the support arms and the feed. The second 2D-plane in turn consisted of the eight measurement points around the dish's rim. The round rim was unfolded so that the accelerometers lay on a straight line rather than a circle circumference. The plotted mode shapes show the accelerometers' original positions as well as the deformed positions.



(a) Test shape of mode 1 (b) FE model shape of mode 1

Figure 6.2: Comparison in the x-y plane



(a) Test shape of mode 1 (b) FE model shape of mode 1

Figure 6.3: Comparison in the x-z plane

6.2.2 Modal Assurance Criteria

As mentioned previously, the MAC is a measure of how well mode shapes match each other. The matrix attained from the calculation of the MAC was plotted as a colour map in order to simplify the comparison. A *Python* script was written to both calculate the MAC and plot the colour map.

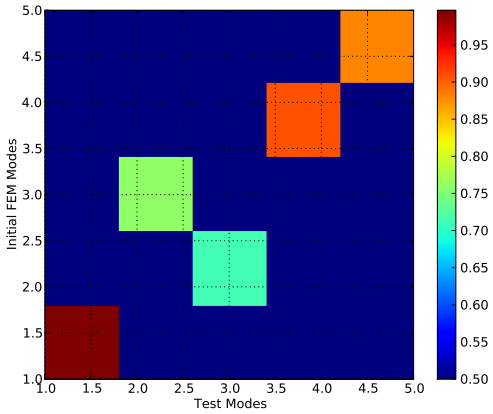
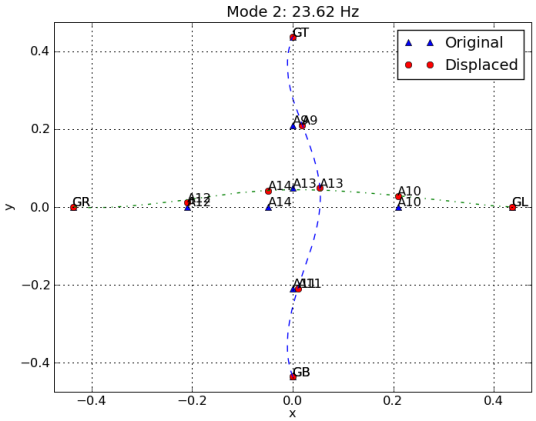


Figure 6.4: MAC of test vs initial FE model's mode shapes

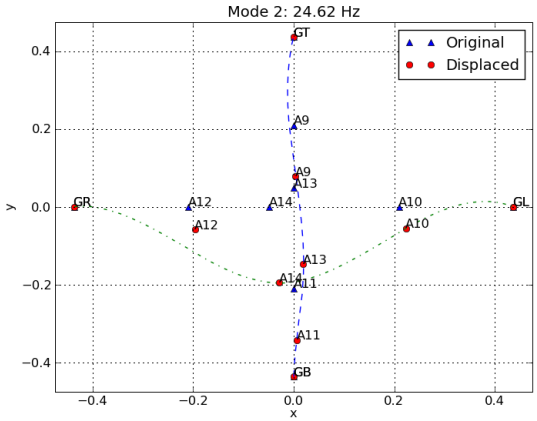
As can be seen from the MAC plot above the 2^{nd} and 3^{rd} modes are in the incorrect order. Figure 6.5 illustrates the discrepancy in the modes.

6.3 Conclusion

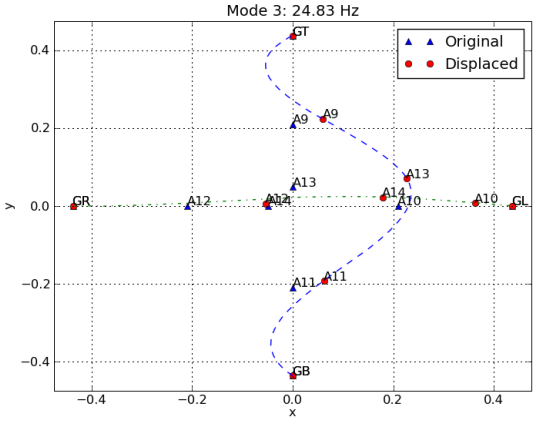
The above comparisons show that the frequencies of the physical and FE model match fairly accurately, with an average error of 4.04 % and a maximum error of 7.99 %. Except for the 2^{nd} and 3^{rd} modes being in the incorrect order, the mode shapes also match fairly accurately. In the optimisation process, the emphasis should consequently be on minimising the error between the frequencies and correcting the order of the 2^{nd} and 3^{rd} mode shapes.



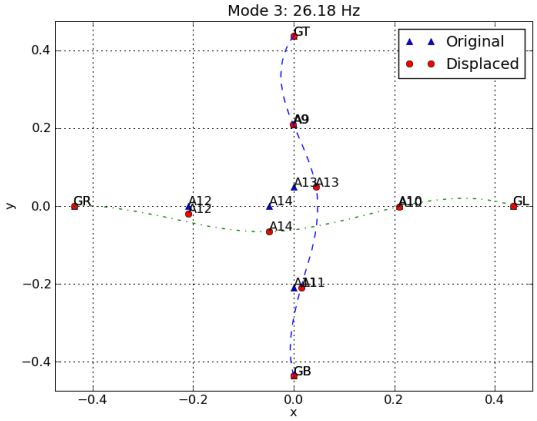
(a) Test shape of mode 2



(b) FE model shape of mode 2

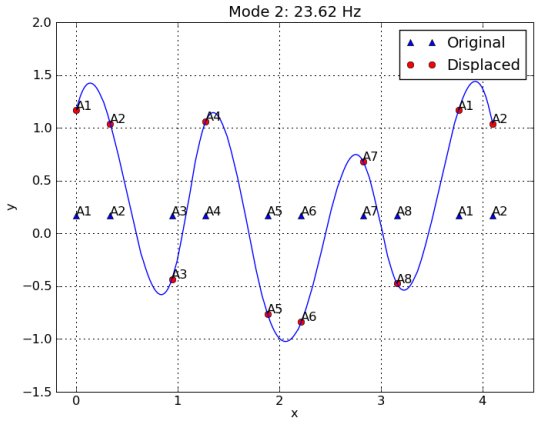


(c) FE model shape of mode 3

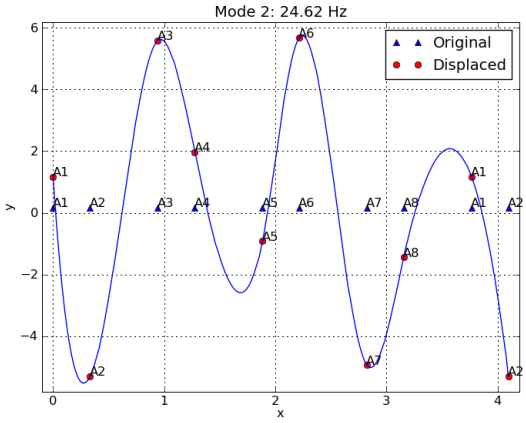


(d) Test shape of mode 3

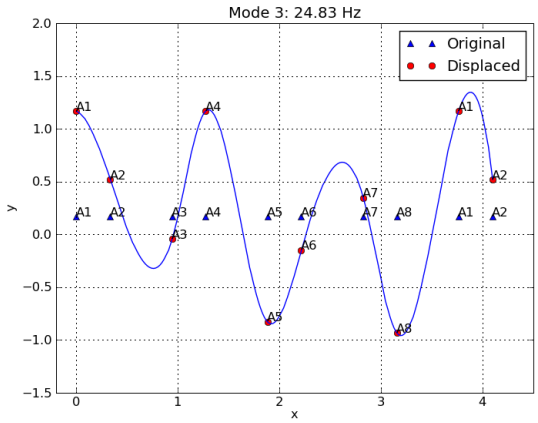
Figure 6.5: x-y plane comparison between modes 2 and 3



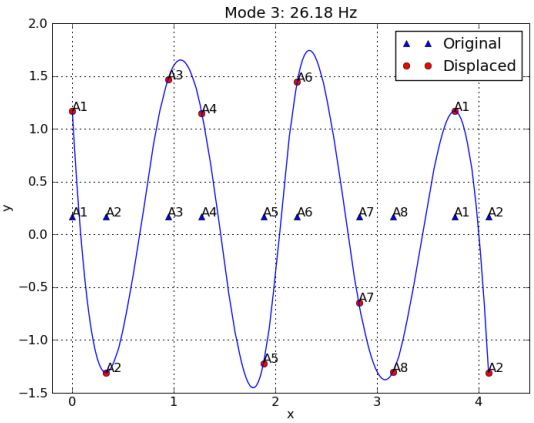
(a) Test shape of mode 2



(b) FE model shape of mode 2



(c) FE model shape of mode 3



(d) Test shape of mode 3

Figure 6.6: x-z plane comparison between modes 2 and 3

Chapter 7

Optimisation

This chapter will cover the methodology that underlies the optimisation process, as conducted in this project to correlate the dynamic characteristics of the FE model with that of the physical model. A discussion will be provided of how the optimisation problem was set up and the specified set-up achieved using *VR&D*'s *GENESIS* software with the *Design Studio* frontend.

7.1 Optimisation Set-up

7.1.1 Objective Function

The objective function was to minimise the least squares error between the mode shapes. Each node that represented an accelerometer position was given a displacement value, i.e. the value measured during the vibration tests. Each node was given 5 displacement values, each value representing the displacement required for one of the 5 modes. Therefore the objective function consisted of 70 terms and all these terms were equally weighted.

7.1.2 Constraints

Inequality constraints were applied to the frequencies, which were given an 8 % error range on the test frequencies. This range was chosen as this was the largest error in frequency for the initial comparison between the numerical and physical models. This error was used to ensure that the frequency error would not be larger than that of the initial match.

The table below illustrates the bounds on the frequencies.

Table 7.1: Constraint definition

Mode	Test Frequency (Hz)	Initial FE Model Frequency (Hz)	Lower Bound (Hz)	Upper Bound
1	13.93	13.05	12.81	15.04
2	23.61	24.62	21.72	25.49
3	26.15	24.83	24.06	28.24
4	32.06	34.62	29.49	34.62
5	42.13	42.75	38.76	45.50

7.1.3 Design Variables

Shape optimisation was used to adjust the shape of the dish. This was done using perturbation vectors that were each assigned their own design variables. The vectors' descriptions follow in Section 7.2, where only the vectors used in the eventual solution is discussed in detail. Sizing optimisation was used to design the element thicknesses that represented the welded part of the dish.

7.2 Methodology

As mentioned in Section 4.2.2, the FE model's output had to be compatible to that of the vibration test. The output request was to gain the displacement values of the specified grid points that represented the accelerometer positions on the physical structure.

The next step was to find explanations for the differences in the data in order to formulate an optimisation problem that would minimise these differences. Two methods were used in an attempt at identifying the differences in the models, namely visual comparison and topometry optimisation.

7.2.1 Visual Comparison

The FE model is based on the designed model, therefore it has no imperfections or any variations from the design. Manufactured models, however, often vary from the intended design. A few differences can therefore be found by visually comparing the two models. The support arms were, e.g., not at an angle of exactly 90^o with respect to one another and did not line up properly.

Another possibility was that the support arms and dish were bent during the assembling of the structure. Although welding will cause a change in the structural properties around welded sections, the FE model does not furthermore have a representation of these sections.

7.2.2 Topometry Optimisation

As mentioned earlier, topometry is a special kind of size optimisation that changes each individual shell element's thickness. Topometry optimisation has no relevance to the problem in engineering terms because the thickness of the dish is constant. It can, however, provide an indication of which parts of the dish have a higher stiffness.

By applying the same objective function and constraints as in the original problem, the optimiser will solve the problem by adjusting the thickness of all the individual elements. Topometry optimisation can be applied to groups of elements in order to decrease the time needed to solve the problem. The result will show the exact stiffness of the structure at any given point and give the user an indication of how to adjust the shape accordingly if required. The analysis showed, e.g., that the elements between two specific support arms had increased in thickness more than elsewhere. This part of the dish therefore probably had a higher stiffness than other sections. The aforementioned resulted in a vector pushing the two arms closer together on the FE model, as measurements on the physical model later confirmed.

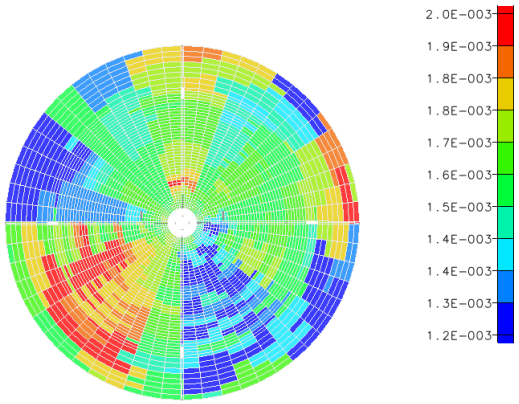


Figure 7.1: Topometry result

7.2.3 Perturbation Vectors

The above two methods were used to identify the vectors needed to change the shape of the FE model. The final solution used 12 shape domains and 13 design variables, of which 12 were associated with perturbation vectors and 1 with the sizing of the weld. The design variables were assigned to 20 vectors - some changes required 3 different vectors that all had the same design variables. Each support arm was given the same type of vectors. Only one arm's vectors will therefore be explained in detail. The following vectors were used in the final optimisation problem:

Support Arm Shift

Two vectors were applied to the support arm in order to shift it perpendicular to its own plane, as shown in Fig. 7.2. The reason for this adjustment was to compensate for the misalignment of the support arms. These vectors however, caused the hole in the centre of the dish to become noncircular. It also caused some elements to become badly deformed. To compensate for this two other vectors (Fig. 7.3) were applied in order to maintain a smooth mesh and to keep the hole in the centre of the dish round.

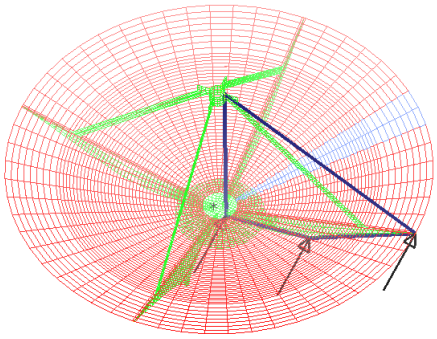


Figure 7.2: Support arm shift

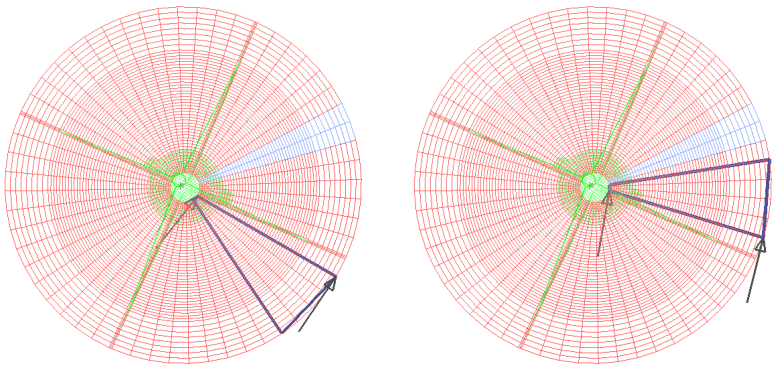
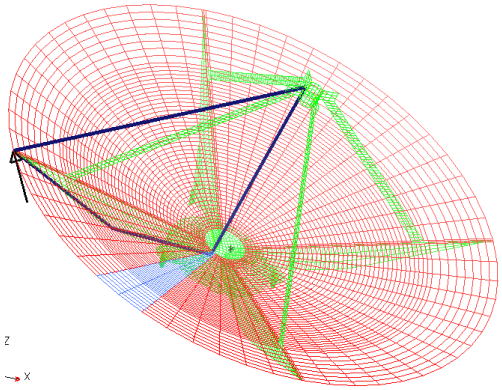


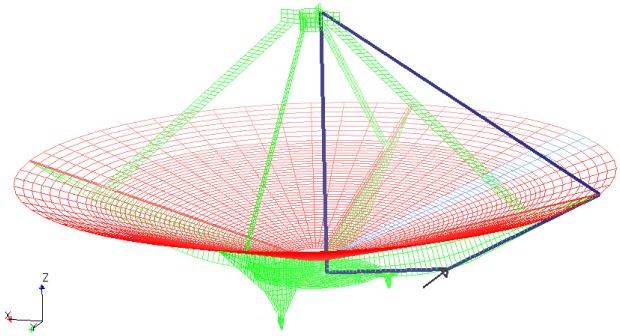
Figure 7.3: Mesh smoothing vectors

Shape Adjustment of Supports

The shape of the support arms was changed using two vectors. These vectors were both applied to the section of the support arm located underneath the dish's surface. The first vector was applied to the tip of the support arm in order to register any bending along the support arm's length. The second vector was applied to the very bottom section of the support arm, half way along its length. This vector was used to register any bending along its height. Both these vectors are illustrated in the figures below.



(a) Adjustments along length



(b) Adjustments along height

Figure 7.4: Shape adjustments of the supports

7.2.4 Initial Values for Design Variables

The initial values for the design variables may partially determine whether an optimum can be found. These values cannot therefore be picked at random. In order to obtain a good starting point, a *Python* script was written to generate 10 random starting points and then let *GENESIS* perform optimisation with each of these starting points. The plot below shows the values of the final objective functions. Not all of these final objective functions' values represent a feasible solution, i.e. one in which all the modes were matched. The final design variables' values for the lowest objective function were then used as initial values for the design variables. This ensured a good starting point for the search for the optimum.

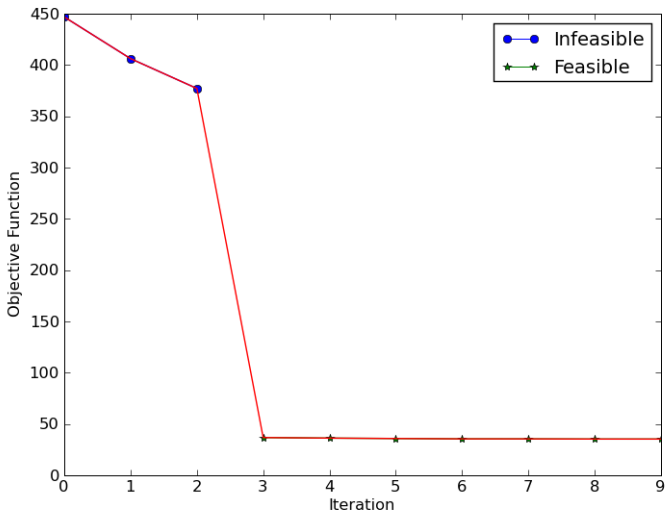


Figure 7.5: Objective function plot

7.3 Results

The vectors adjusted the support arms so that they were no longer perfectly aligned or perfectly straight. The figures below provide a comparison of the original FE model with the updated model.

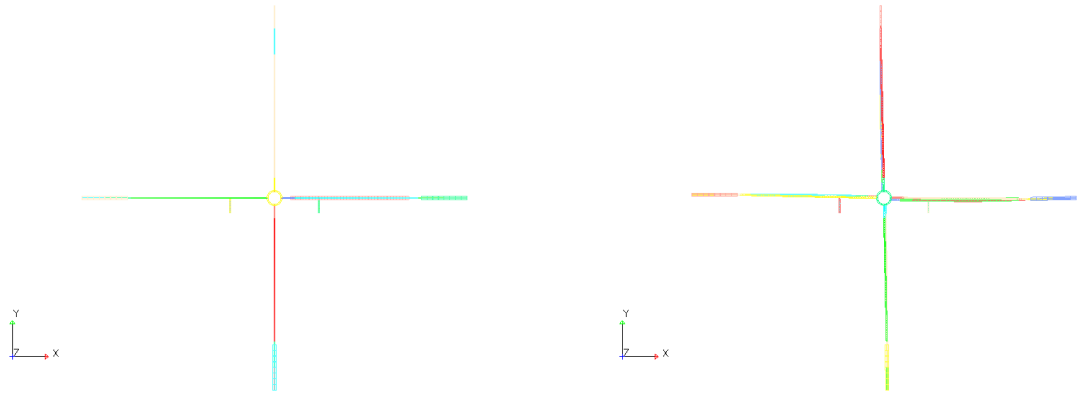


Figure 7.6: Comparison in the x-y plane

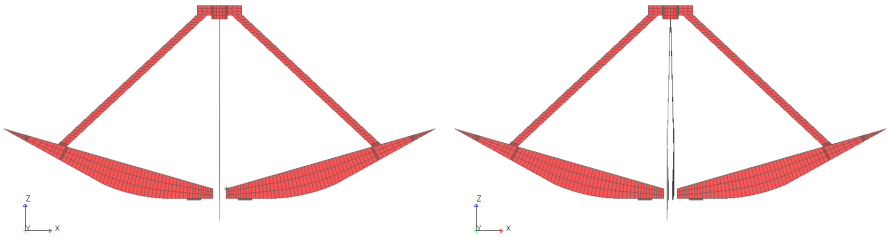


Figure 7.7: Comparison in the x-z plane

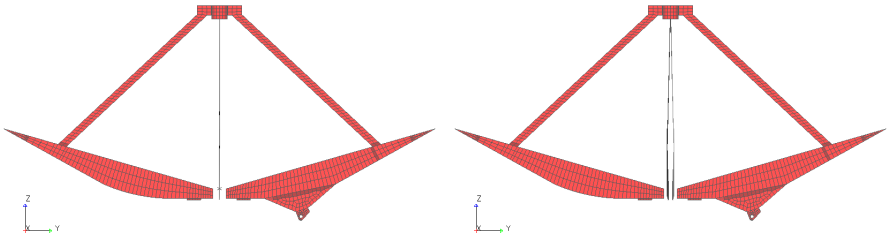


Figure 7.8: Comparison in the y-z plane

The thickness of the welded section was increased from 1.6 mm to 1.68 mm. This was such a small alteration that it had no effect on the shape correlation and only a slight influence on the frequency match. Analysis without the weld gave an average error in the frequencies of 2.93 %, whilst analysis with the weld gave an average error of 2.77 %.

The frequencies and the mode shapes match was compared in the same way as in Chapter 6. The results are provided in the following figure and table.

Table 7.2: Frequency comparison between test and updated FE model's data

Mode	Test Frequency (Hz)	Updated FE Model Frequency (Hz)	Error (%)	MAC
1	13.93	13.77	1.65	0.996
2	23.61	23.96	0.64	0.892
3	26.15	25.58	2.60	0.913
4	32.06	34.60	7.89	0.912
5	42.13	42.59	1.23	0.892
Average			2.80	0.921

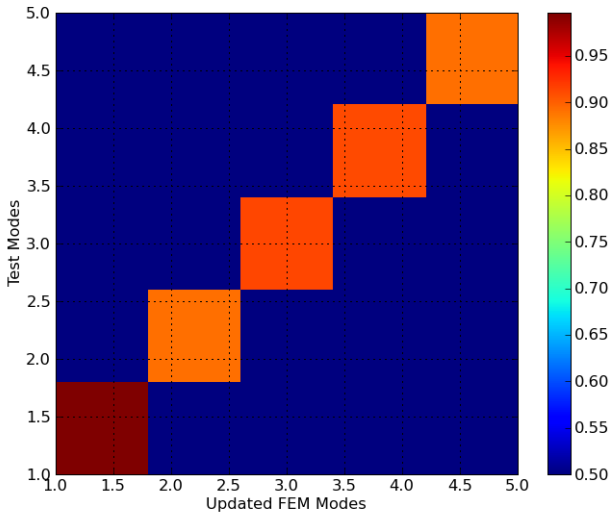


Figure 7.9: MAC plot of test vs updated FE model

7.4 Conclusion

The matching process allowed the updated FE model to be a better representation of the physical model. Whilst the order of the 2^{nd} and 3^{rd} modes was corrected, the correlation of all the modes as well as the overall frequency match was improved. The frequency improvement was from an average error of 4.04 % to 2.80 %, while the average MAC value was improved from 0.647 to 0.921.

Through optimisation, the updating of the FE model with test data was successfully completed. The aforementioned allowed for the automation of the whole process as well as the simultaneous matching of all the modes. The latter would have been difficult to achieve using traditional model updating methods.

Although the optimisation techniques were only applied to a relatively simple model, they could be applied to larger, more complex models, such as the actual KAT-7 antenna structure.

As optimisation allows for the automation and simultaneous matching of all required modes, optimisation techniques may be of significant value to an engineer attempting model updating. Not only will it save time, it will also allow for the use of more detailed FE models than those allowed by other methods. The more detailed the FE model is, the more likely an analyst will gain a detailed understanding of the structure's response.

Chapter 8

Validation Tests

The data from the numerical and physical model were successfully correlated. The set-up in which the tests were carried out will, however, be very complex to recreate when working on the actual KAT-7 antenna structure. Removing the antenna from its support structure and performing the physical tests under 'free-free' support conditions will be a challenging task. In order to validate the results achieved from the previous correlation, another set of tests was carried out in the same manner as that which would be needed for the KAT-7. Here, the vibration tests were performed with the dish attached to its support structure, which was in turn fixed to a grounded structure, simulating the true set-up if these tests were to be performed on the KAT-7.

8.1 Vibration Tests

The vibration tests were performed in the same manner in the validation tests as in the original tests. The only difference was in the manner in which the dish was supported. The validation tests were performed with the dish attached to its support structure, which was in turn bolted to a solid base (see Fig. 8.1). The accelerometers were placed in the same configuration as in the original tests so that the data could be compared. The exciter's excitation point was also kept in the same place; although it was now suspended from bungee cords instead of being fixed to the ground. The procedure for performing the tests as well as acquiring the data was the same as that used in the original tests.

Figure 8.1: Validation test set-up

8.2 Data Comparison

The data comparison focused on data from the validation tests and from a FE model of the physical structure in the same set-up as that used in the validation tests, i.e. the FE model represented the dish attached to its support structure, that was in turn fixed at its base. As done during the initial data correlation, the first 5 modes were compared by looking at each mode's frequency and shape. Firstly, a comparison was done between the validation test data and that of the original FE model. Next, a comparison of the validation test data and that of the updated model (optimised FE model) was performed in order to establish whether the updated model is an improved representation of the physical structure or not. The results of the two comparisons follow below.

Table 8.1: Frequency comparison of the validation tests with the initial FE model

Mode	Test	Initial FE model	Error	MAC
1	13.87 Hz	13.05 Hz	5.91 %	0.642
2	16.63 Hz	16.32 Hz	1.87 %	0.514
3	28.71 Hz	28.62 Hz	0.30 %	0.813
4	32.40 Hz	35.51 Hz	9.59 %	0.893
5	42.61 Hz	42.97 Hz	0.84 %	0.281
Average			3.70 %	0.628

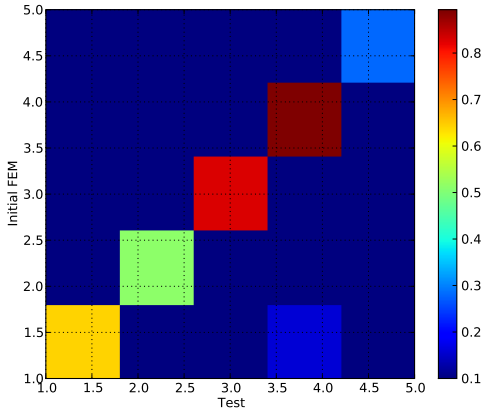


Figure 8.2: MAC of validation test vs initial FE model

Table 8.2: Frequency comparison of the validation tests with optimised FE model

Mode	Test	Optimised FE model	Error	MAC
1	13.87 Hz	13.37 Hz	3.60 %	0.818
2	16.63 Hz	16.74 Hz	0.66 %	0.880
3	28.71 Hz	28.60 Hz	0.37 %	0.773
4	32.40 Hz	35.39 Hz	9.21 %	0.912
5	42.61 Hz	42.56 Hz	0.13 %	0.268
Average			2.79 %	0.730

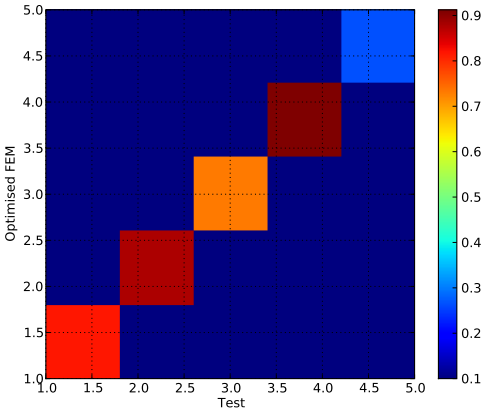


Figure 8.3: MAC of validation test vs optimised model

8.3 Conclusion

From the results of the comparisons supplied in the previous section, it is evident that there was a general improvement in both the frequencies' and mode shapes' match of the updated model. The only frequency that was not improved was that of mode 3, although the difference in the frequencies is only 0.02 Hz. The mode shapes that were not improved were the 3^{rd} and 5^{th} modes. These modes, especially the 5^{th} mode, have deformations in the rod where the other modes do not. Another aspect that may account for the lack of improvement is that the pedestal was not included in the model updating process. As seen in Section 4.2.3, the pedestals effect on the frequencies is minimal, however it has an effect on the mode shapes as seen from Fig. 8.2. This plot shows that the initial FE model's modes are in the correct order where as the dish's initial FE model's modes are not. This could be a reason for the mismatch between the numerical and physical model as the rod and the pedestal were not part of the structure during the optimisation process. This could also be the reason for the generally poor match between the 5^{th} modes.

Chapter 9

Conclusion

This chapter will provide the final conclusions of this study and make a few suggestions for future work in the relevant field.

9.1 Overview

The goal of this study was to illustrate how structural design optimisation can assist an engineer when faced with an experimental data matching problem. This was done using a simplified, scaled model of the KAT-7 antenna structure. This physical model underwent vibration testing in order to gain its modal frequencies and their corresponding mode shapes. The dish structure was not attached to its pedestal during testing as FE analysis showed that the pedestal did not have a large effect on the frequency range of interest.

A FE model of the physical model was generated and analysed in order to gain its modal frequencies and mode shapes. Hereafter, the frequencies and mode shapes of the two sets of data were compared. The initial comparison revealed some discrepancies between the two sets - the frequencies had an average error of 4.04 % and a maximum error of 7.99 %. The mode shapes were compared using a number of techniques, but the most quantifiable technique was the use of the modal assurance criteria. The average MAC value for the initial comparison was 0.647. A MAC plot furthermore revealed that the 2^{nd} and 3^{rd} modes were in the incorrect order in the FE model.

The differences between the models' data were investigated by visually comparing the two models and by employing topometry optimisation. This investigation revealed that the discrepancies were mainly due to the orientation

and shape of the support arms on the dish structure.

An optimisation problem was set up in order to improve the frequencies' and mode shapes' match. The set-up was based on an objective that was used to minimise the least squares error of the mode shapes. Frequency constraints were imposed with an 8 % error range on the test frequencies. This was done in order to ensure that the frequency match did not decrease whilst the modes were matched. Design variables were assigned to perturbation vectors that adjusted the shape and orientation of the support arms on the dish structure. One design variable was assigned to sizing optimisation of the welded section of the dish, but this seemed to have a minimal effect.

Once the FE model was updated through optimisation techniques, its data was compared to the test data in order to establish whether it was an improved representation of the physical model. This comparison was performed in the same manner as the initial comparison. The frequency match showed an average error of 2.80 % with a maximum error of 7.89 %. The mode shapes' average MAC value was improved to 0.921 and the order of the 2^{nd} and 3^{rd} modes were corrected.

In order to confirm that the updated model was a better representation of the physical model, a validation test was completed. This test was performed with the dish structure attached to its pedestal, which was in turn fixed to the ground at its base. This would be the required set-up for tests on the actual KAT-7.

Comparisons were again done between test data and the initial and updated FE models. These comparisons revealed that the updated model was a better representation of the physical structure. The average frequency error was improved from 3.70 % to 2.79 %, whilst the average MAC value was improved from 0.628 to 0.730.

Although there was a general improvement in the data match for the structure as a whole, the 5^{th} mode did not improve. This may be attributed to the pedestal and the rod not being included in the optimisation process.

Optimisation allowed for an improved FE model of the physical structure. The updating process was automated and allowed for all the modes to be matched simultaneously. Automation and simultaneous matching of modes has previously been very difficult to achieve.

9.2 Future Work

Although this project was based on a simplified, scaled model of the KAT-7, the techniques could be applied to the actual structure. Optimisation could be used to update the FE model of the KAT-7 and improve the confidence in the model. This would be particularly advantageous to the KAT-7 project as it is part of a much larger project, the MeerKAT, which will consist of 80 antennas.

Applying optimisation techniques to a large and complex model such as the KAT-7 would serve to test the potential of optimisation and to identify any of its limitations with respect to experimental data matching.

Bibliography

Allemang, R. (2003). The modal assurance criteria - twenty years of use and abuse. Sound and Vibration/August 2003. 15

Bauermeister, H. and More, L. (2008). Kat-7 antenna structure design and analysis. Tech. Rep., MMS Technology (Pty)Ltd, Centurion, RSA. viii, 25, 29, 30, 31, 32

Candan, S. (2000). Shape optimization using abaqus and visualdoc. *Journal of AIAA*. 19, 20

Dascotte, E. and Strobbe, J. (1998). *Updating Finite Element Models using FRF Correlations Functions*. Dynamic Design Solutions (DDS) NV, Interleuvenlaan64, B-3001 Leuven, Belgium. 14, 23

Ewins, D. (1984). *Modal Testing: Theory and Practise*. 2nd edn. Research Studies Press Ltd. viii, 7, 8, 9, 10, 13, 14

FEAdomain (2007). Using modal effective mass to determine modes for frequency response analysis. Available at: `http://feadomain.com/e107_plugins/content/content.php?content.76`, [2008, October 19]. 14, 15

Formenti, D. (1999). What is the coherence function and how can it be used to find measurement and test setup problems? Sound and Vibration/December 1999. 13

Friswell, M. and Mottershead, J. (1993). Model updating in structural dynamics. *Journal of Sound and Vibrations*. 23

GENESIS (2008). *Genesis design manual, version 10.1*. Vanderplaats Research and Development, Inc, Colorado Springs. viii, 16, 19, 21, 22, 23

Haystack Radio Telescope [Online] (2005). Available at: `http://www.haystack.mit.edu/edu/undergrad/materials/slide17.GIF`, [2008, October 22]. viii, 7

Inman, D. (2001). *Engineering Vibration.* 2nd edn. Prentice Hall International, Inc. x, 10, 11, 12, 13, 48

Irvine, T. (2000). An introduction to frequency response functions. Available at: `http://www.cs.wright.edu/~jslater/SDTCOutreachWebsite/intro_freq_resp_functions.pdf`, [2008, October 22]. 12

Irvine, T. (2009). *Effective Modal Mass and Modal Participation Factors.* Shock and Vibration Analysis at Vibrationdata, Chandler, Arizona, USA. 15

KAT-7 Update [Online] (2009). Available at: `http://www.ska.ac.za/newsletter/issues/10/03.php`, [2009, July 19]. viii, 26

Leiva, J. (2007). *Methods for generating perturbation vectors for topography optimization of structures.* Vanderplaats Research and Development, Inc, MI, USA. 23

Leiva, J. (2008). *Topometry optimization: a new capability to perform element by element sizing optimization of structures.* Vanderplaats Research and Development, Inc, MI, USA, 2nd edn. 21, 22

Leiva, J. and Watson, B. (1999). *Shape Optimization in the GENESIS Program.* VMA Engineering. viii, 19, 20, 21

MeerKAT Construction [Online] (2009). Available at: `http://www.ska.ac.za/meerkat/overview.php`, [2010, February 19]. 2

Meirovitch, L. (2001). *Fundamentals of Vibration.* International edition edn. McGraw-Hill. 11

Mundt, C. and Quinn, G. (2005). *Test-analysis correlation with design optimisation.* Long Beach Convention Center, Long Beach, CA. 3

PCB [Online] (2009). Available at: `http://www.pcb.com/techsupport/tech_accel.php`, [2010, February 19]. 11

Schmit, L. and Miura, H. (1976). Approximation concepts for efficient structural synthesis. *NASA CR-2552*. 15

SKA, 2008 (2008). *Ready to host the SKA*. Johannesburg, South Africa. viii, 1, 2

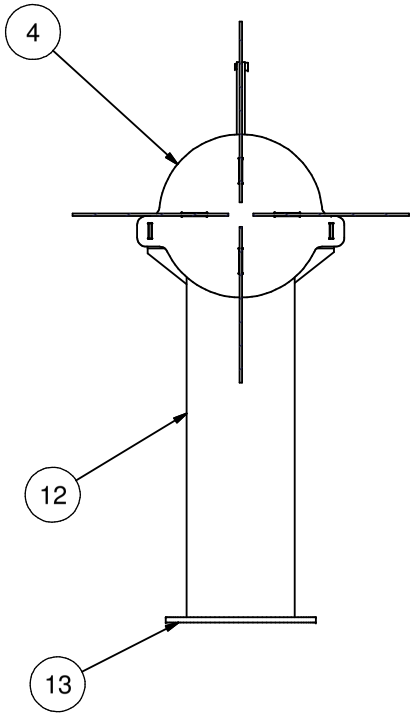
SKA Project [Online] (2008). Available at: `http://www.ska.ac.za/bid/project.php`, [2008, October 22]. 1

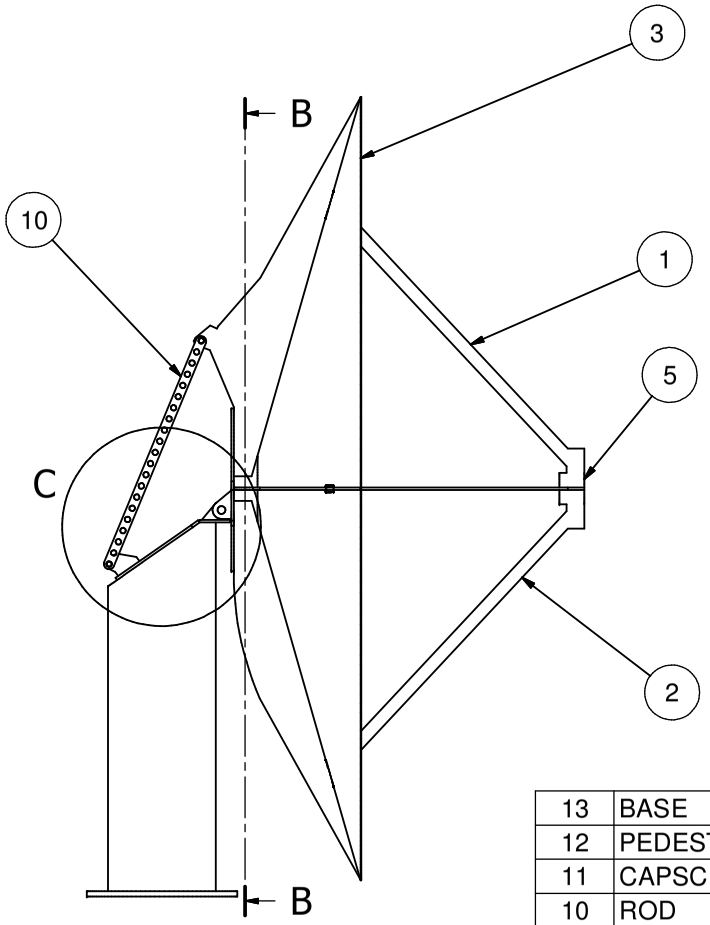
Vanderplaats, G. (2007). *Multidiscipline Design Optimization*. 1st edn. Vanderplaats Research and Development, Inc. 2, 15, 16, 17, 18

Appendix A

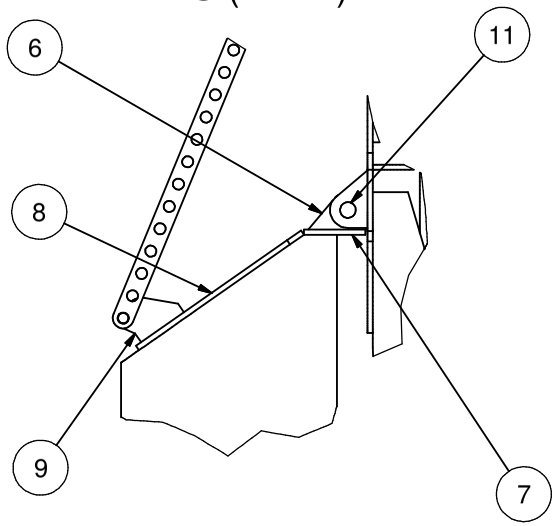
Drawings

B-B (1:8)





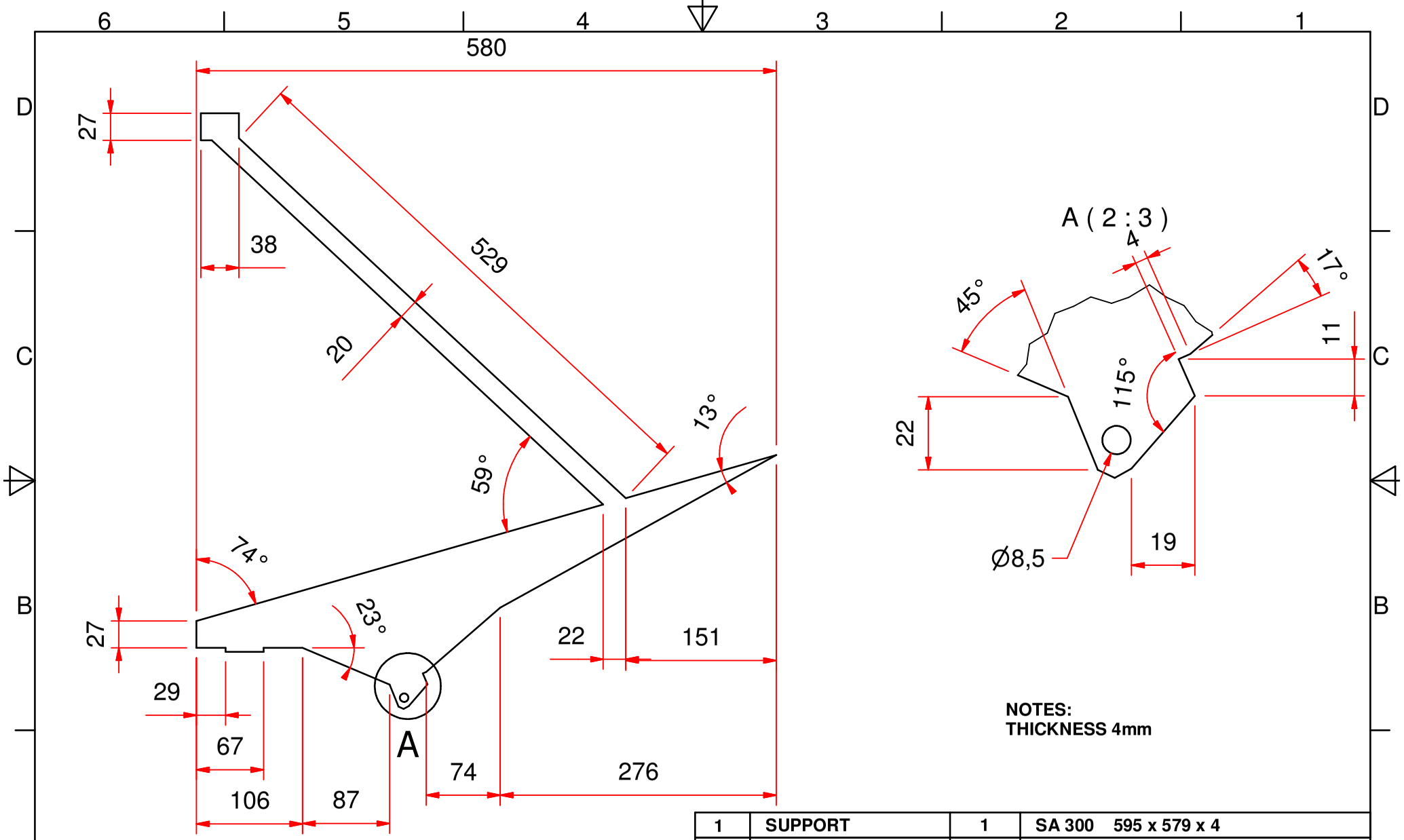
C (1 : 4)



ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
13	BASE	1	ANT - 12
12	PEDESTAL	1	ANT - 11
11	CAPSCREW	4	M8
10	ROD	2	ANT - 10
9	PIVOT	1	ANT - 09
8	YOKE 1	1	ANT - 08
7	YOKE 2	1	ANT - 07
6	ELBOW	4	ANT - 06
5	FEED	1	ANT - 05
4	BACK	1	ANT - 04
3	DISH	1	ANT - 03
2	SUPPORT GEN	3	ANT - 02
1	SUPPORT	1	ANT - 01

UNIVERSITEIT VAN STELLENBOSCH

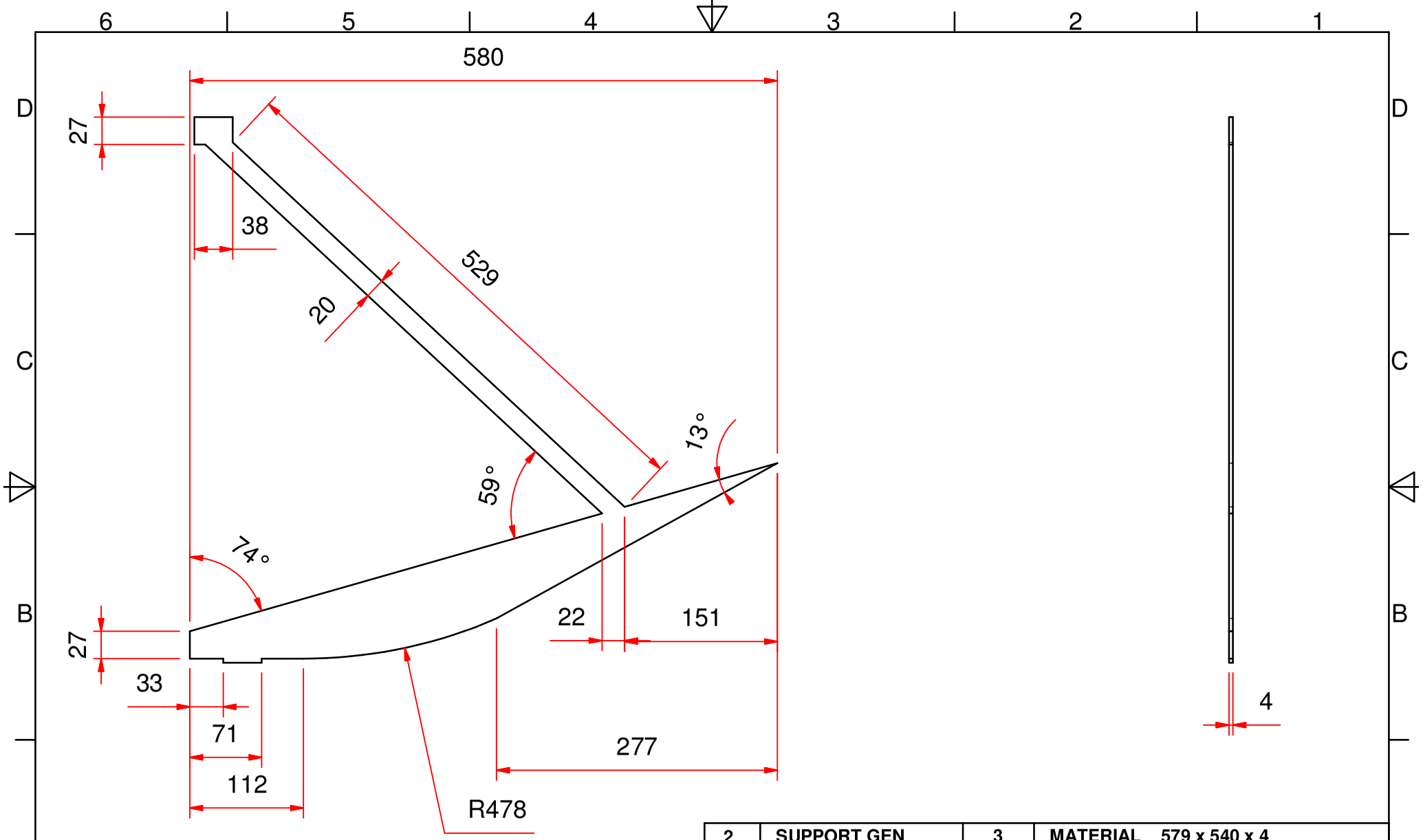
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	SKAAL OP A 1:8 MATE IN mm DATUM 29/09/08	TITEL: SIMPLIFIED MODEL VEL No. 1 VAN 14 VELLE	No.



ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
1	SUPPORT	1	SA 300 595 x 579 x 4

UNIVERSITEIT VAN STELLENBOSCH	SKAAL OP A 1:5	TITEL: SUPPORT	
	MATE IN mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	DATUM 29/09/08
VEL No. 2 VAN 14 VELLE	No. ANT - 01		

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

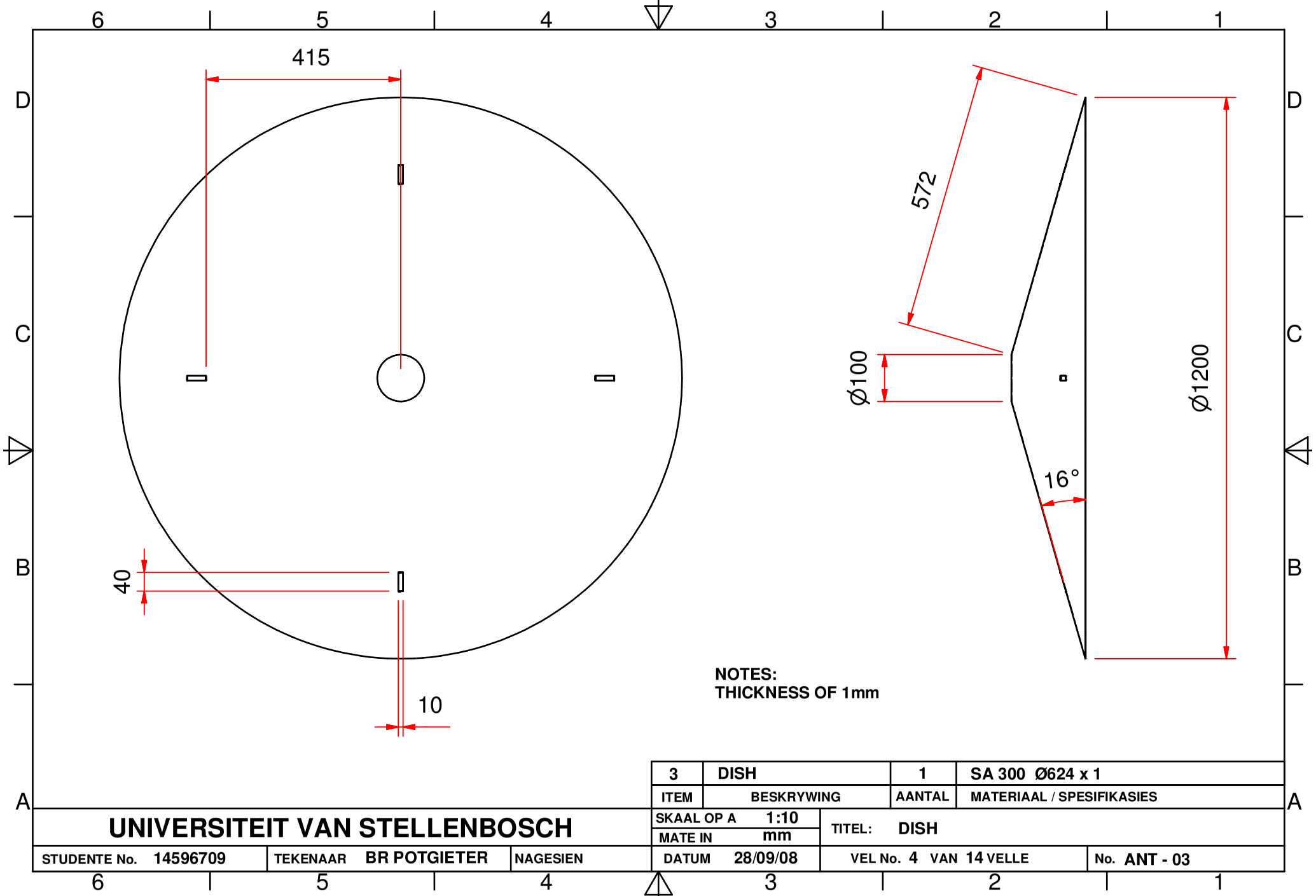


2	SUPPORT GEN	3	MATERIAL 579 x 540 x 4
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A	1:5	TITEL:	GENERAL SUPPORT
MATE IN	mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	
DATUM 29/09/08	VEL No. 3 VAN 14 VELLE	No. ANT - 02	

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

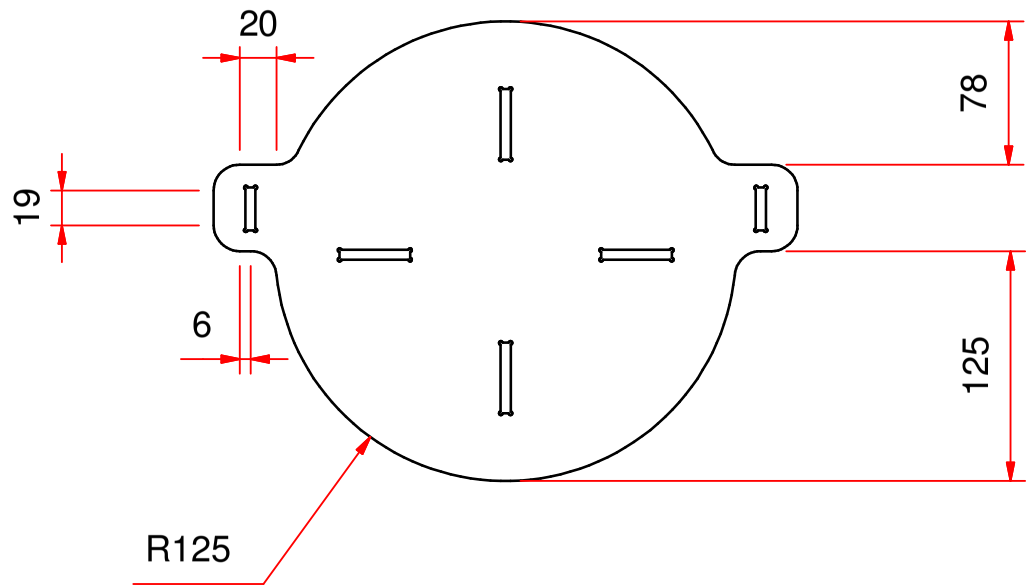


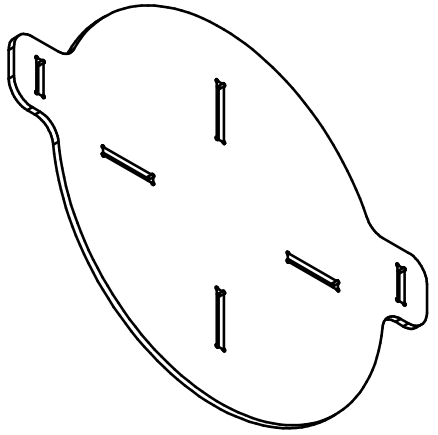
3	DISH	1	SA 300 Ø624 x 1
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

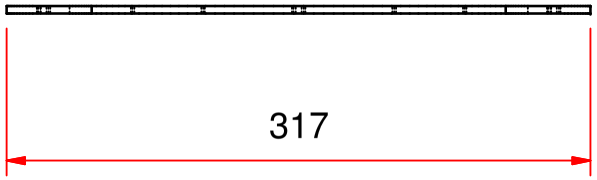
UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A	1:10	TITEL:	DISH
MATE IN	mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	
DATUM	28/09/08	VEL No. 4 VAN 14 VELLE	No. ANT - 03

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





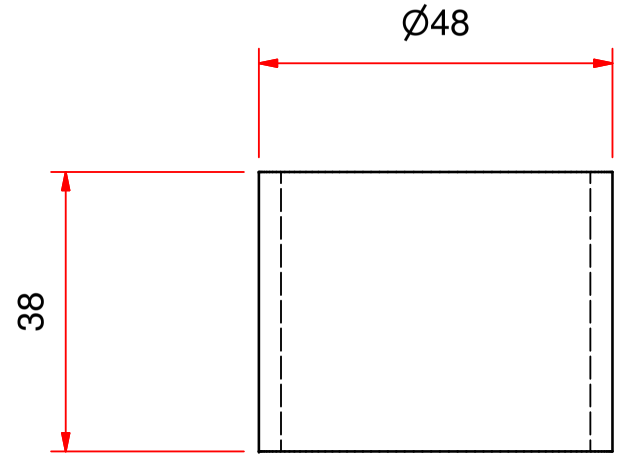


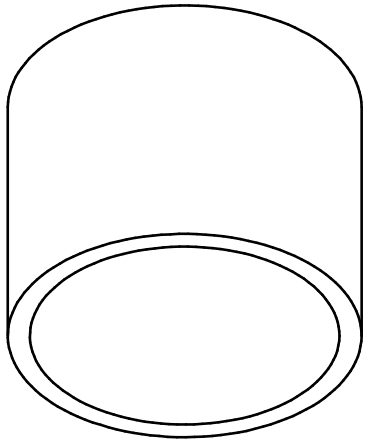
4	BACK	1	SA 300 317 x 250 x 4
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

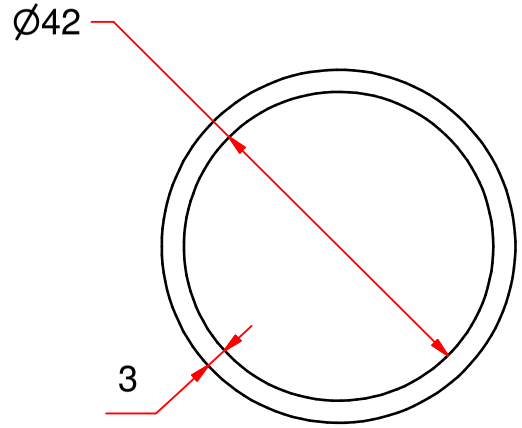
UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A	1:4	TITEL:	BACKING PLATE
MATE IN	mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	
DATUM 29/09/08	VEL No. 5 VAN 14 VELLE	No. ANT - 04	

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





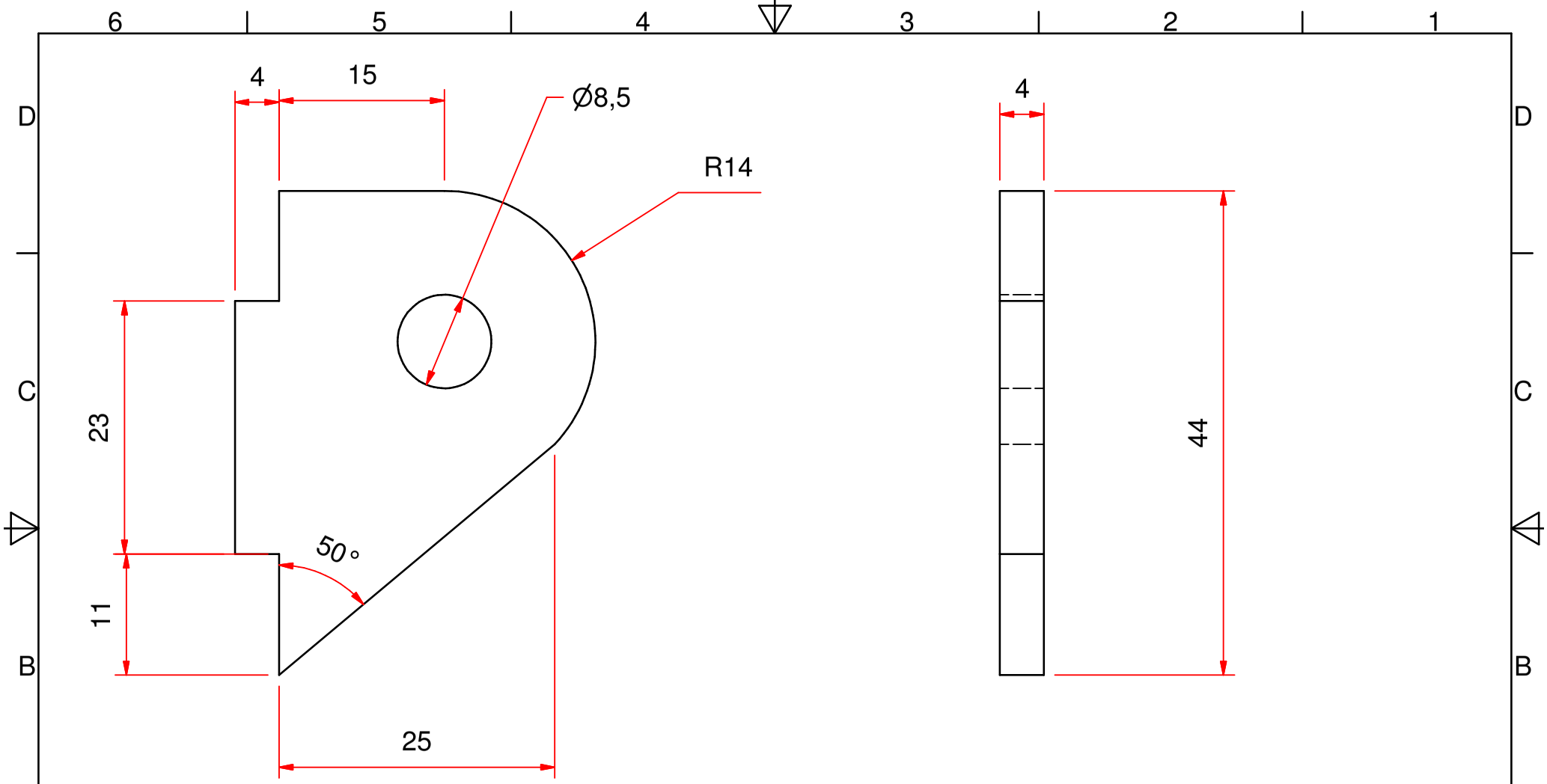


5	FEED	1	SA 300 Ø48 x 38
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A	1:1	TITEL:	FEED
MATE IN	mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	
DATUM 29/09/08	VEL No. 6 VAN 14 VELLE	No. ANT - 05	

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

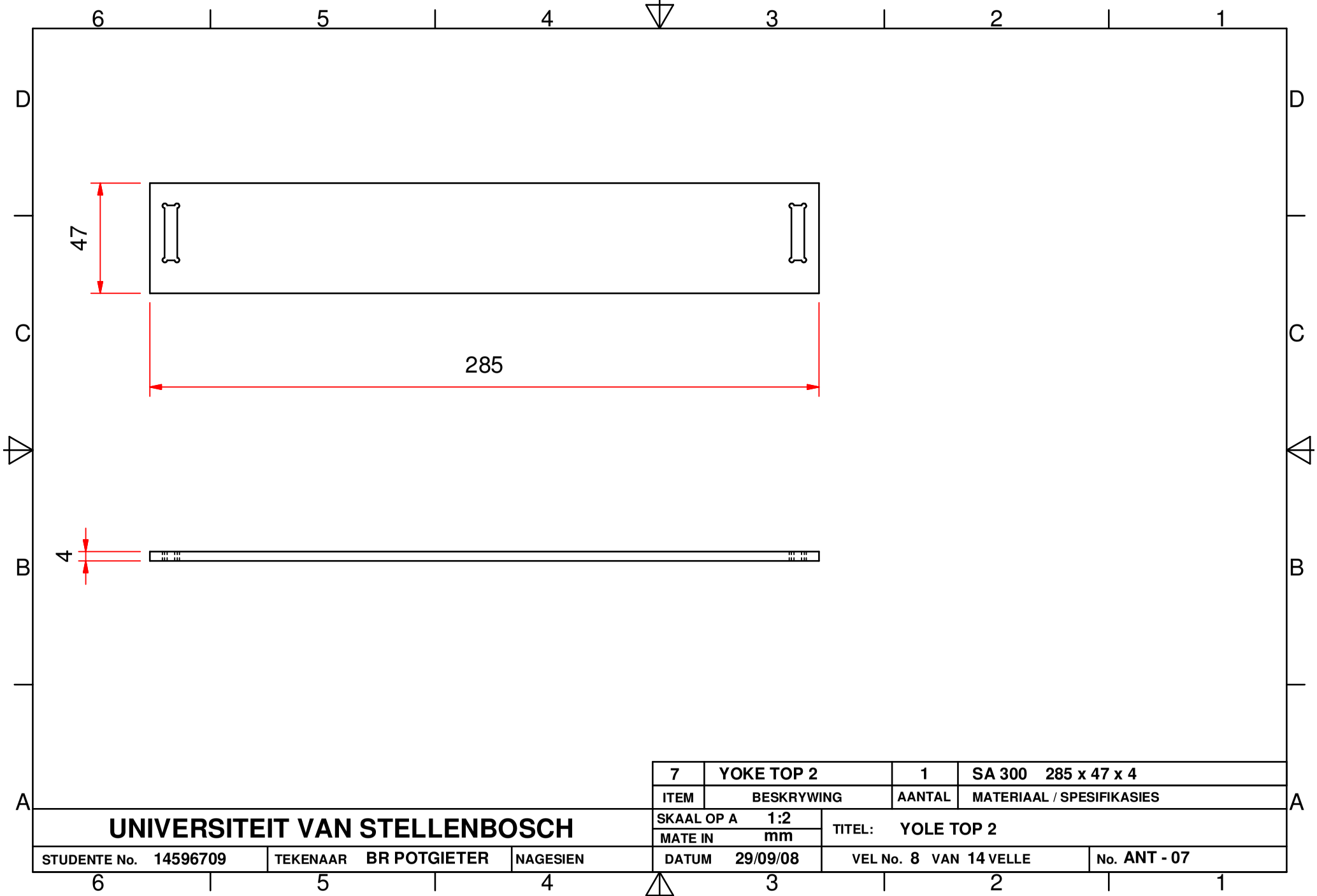


ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
6	YOKE EL	4	SA 300 44 x 35 x 4

UNIVERSITEIT VAN STELLENBOSCH

STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	SKAAL OP A 2:1 MATE IN mm DATUM 29/09/08	TITEL: YOKE EL VEL No. 7 VAN 14 VELLE	No. ANT - 06

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

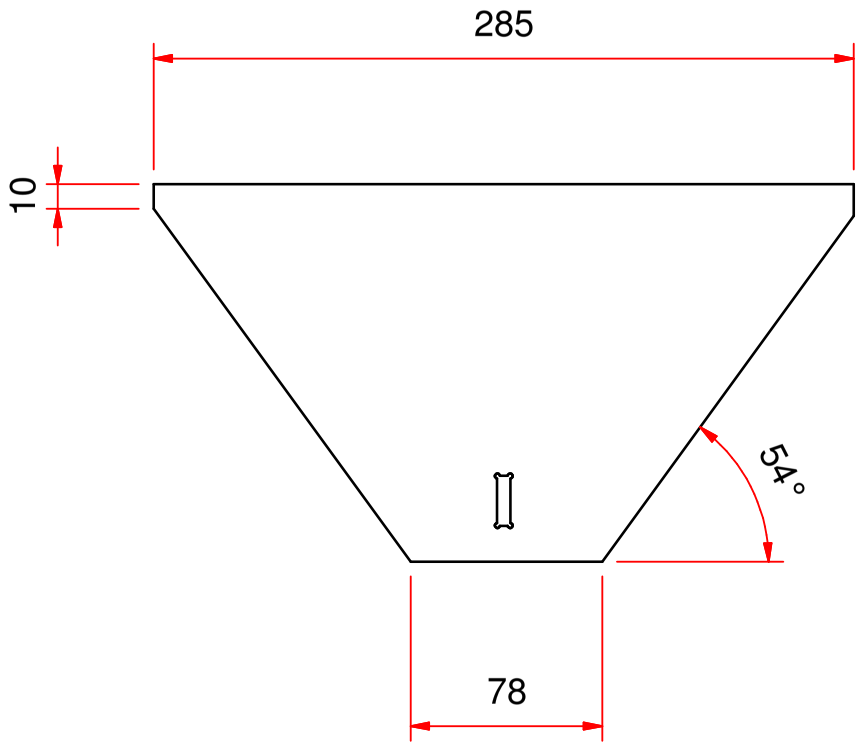


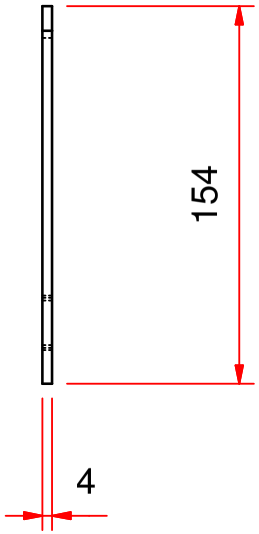
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
7	YOKE TOP 2	1	SA 300 285 x 47 x 4

UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A	1:2	TITEL:	YOLE TOP 2
MATE IN	mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	
DATUM 29/09/08	VEL No. 8 VAN 14 VELLE	No. ANT - 07	

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT



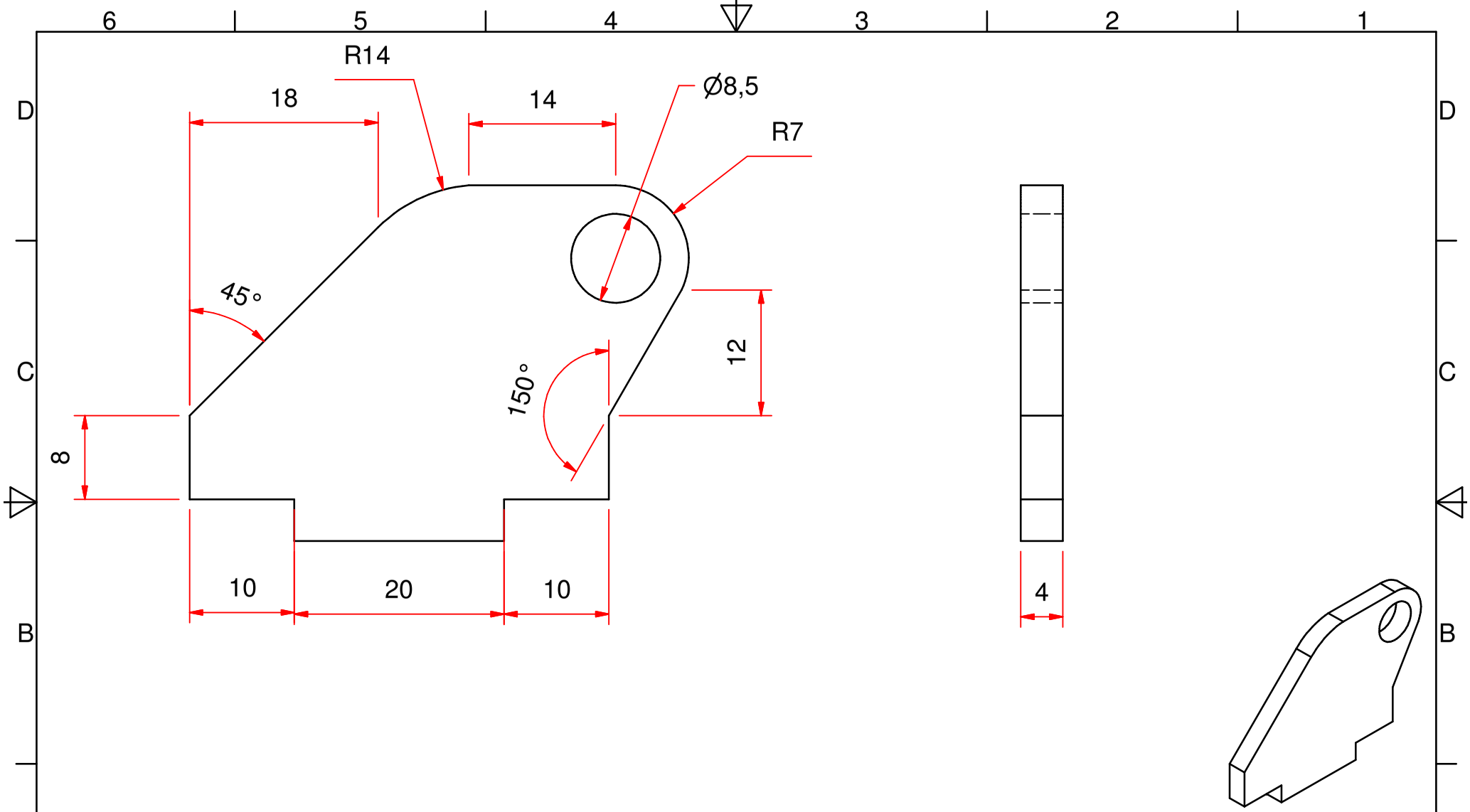


ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
9	YOKE TOP 1	1	SA 300 285 x 154 x 4

UNIVERSITEIT VAN STELLENBOSCH

STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	SKAAL OP A 1:3 MATE IN mm DATUM 29/09/08	TITEL: YOKE TOP 1 VEL No. 9 VAN 14 VELLE	No. ANT - 08

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

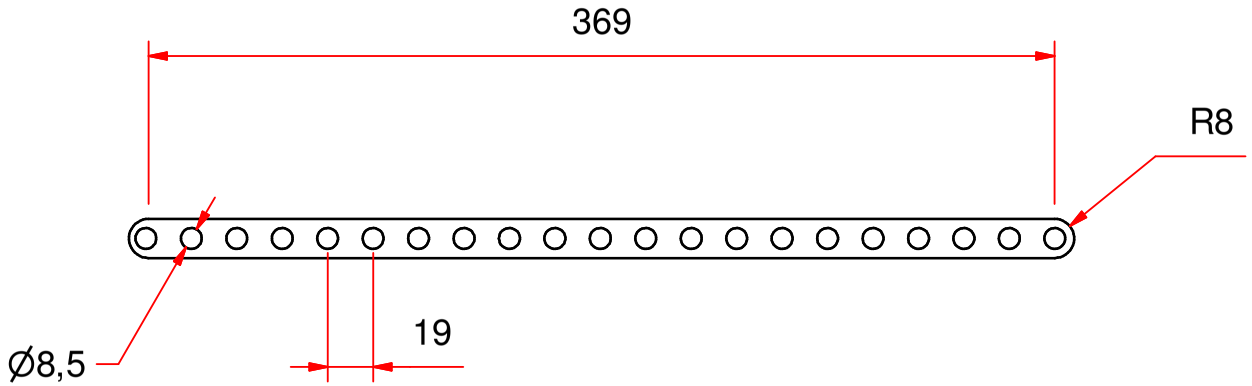


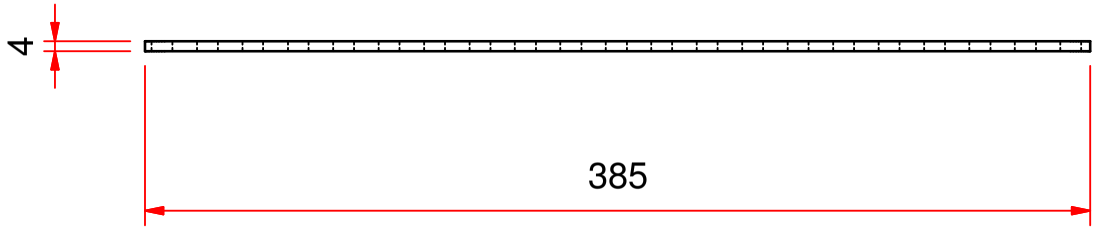
14	PIVOT	1	SA 300 48 x 34 x 4
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

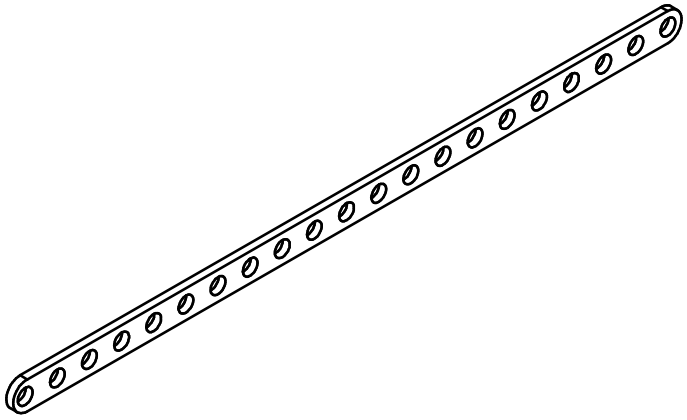
UNIVERSITEIT VAN STELLENBOSCH

SKAAL OP A 2:1	TITEL: PIVOT	
MATE IN mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN
DATUM 29/09/08	VEL No. 10 VAN 14 VELLE	No. ANT - 9

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





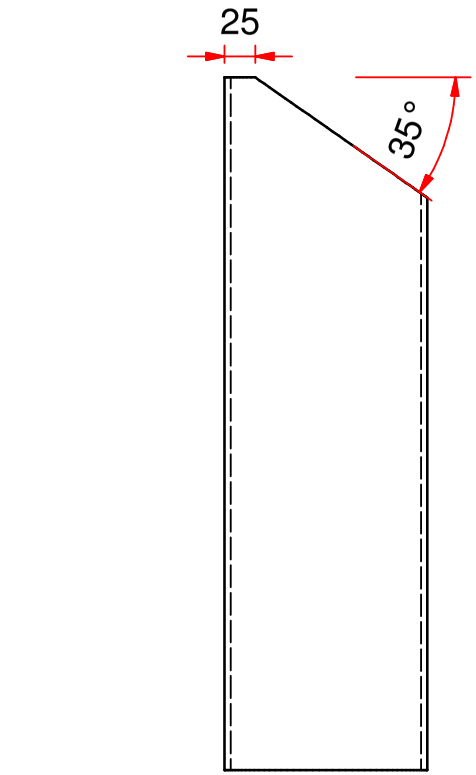


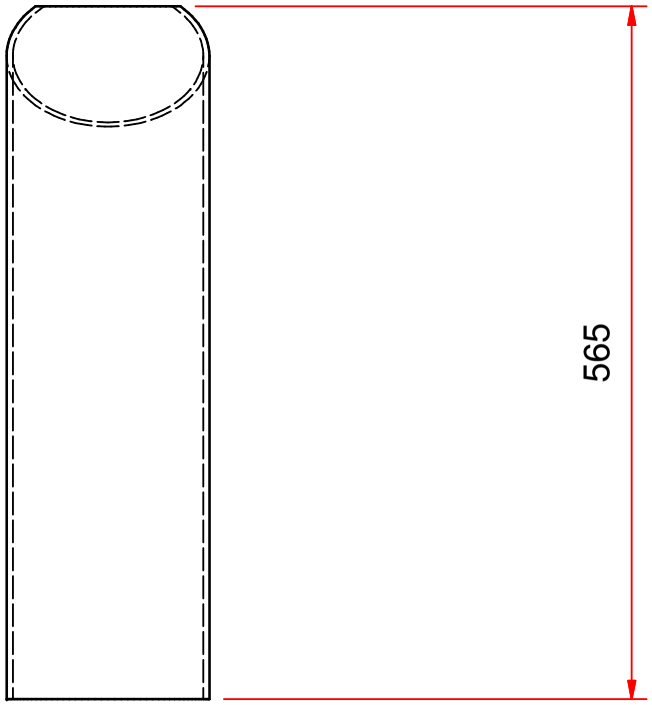
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
15	SCREW	2	SA 300 385 x 16 x 4

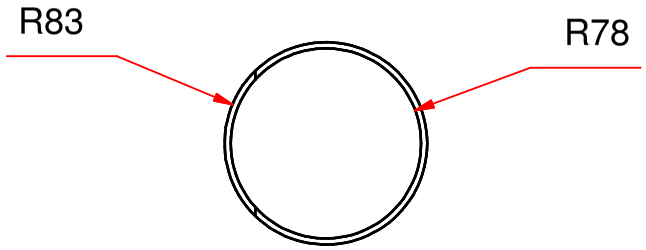
UNIVERSITEIT VAN STELLENBOSCH

STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	SKAAL OP A 1:3 MATE IN mm DATUM 29/09/08	TITEL: SCREW VEL No. 11 VAN 14 VELLE	No. ANT - 10

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





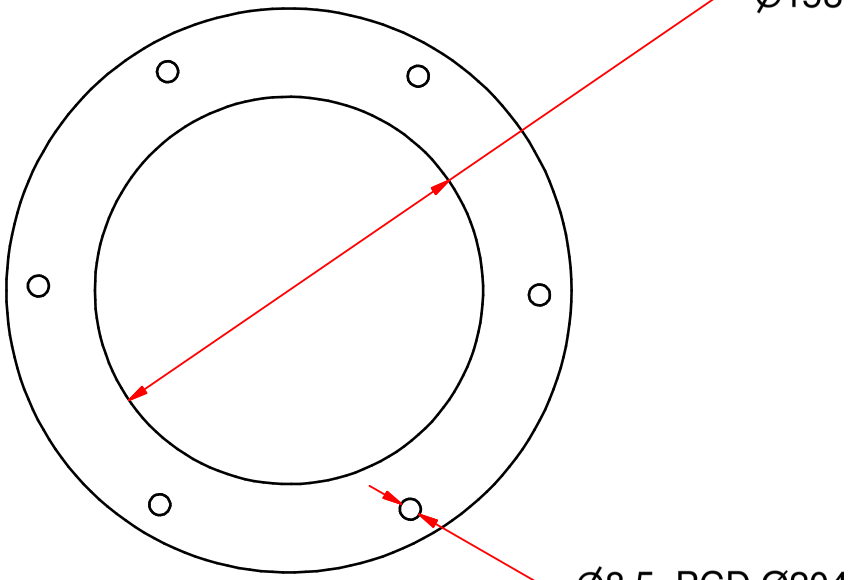


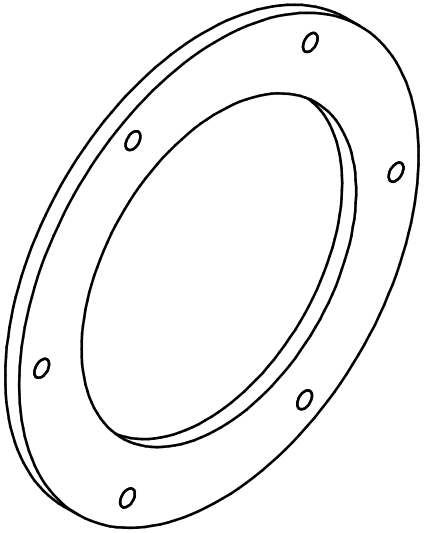
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES
1	PEDESTAL	1	SA 300 Ø166 x 565

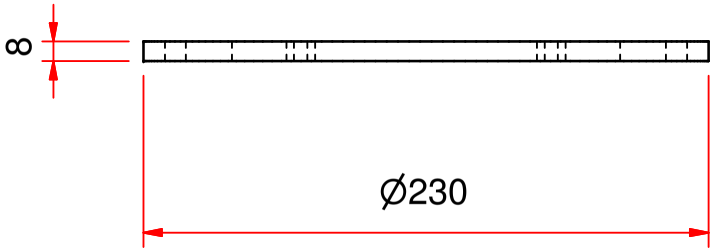
UNIVERSITEIT VAN STELLENBOSCH

STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	SKAAL OP A 1:6 MATE IN mm DATUM 01/01/08	TITEL: PEDESTAL VEL No. 12 VAN 14 VELLE	No. ANT - 11

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





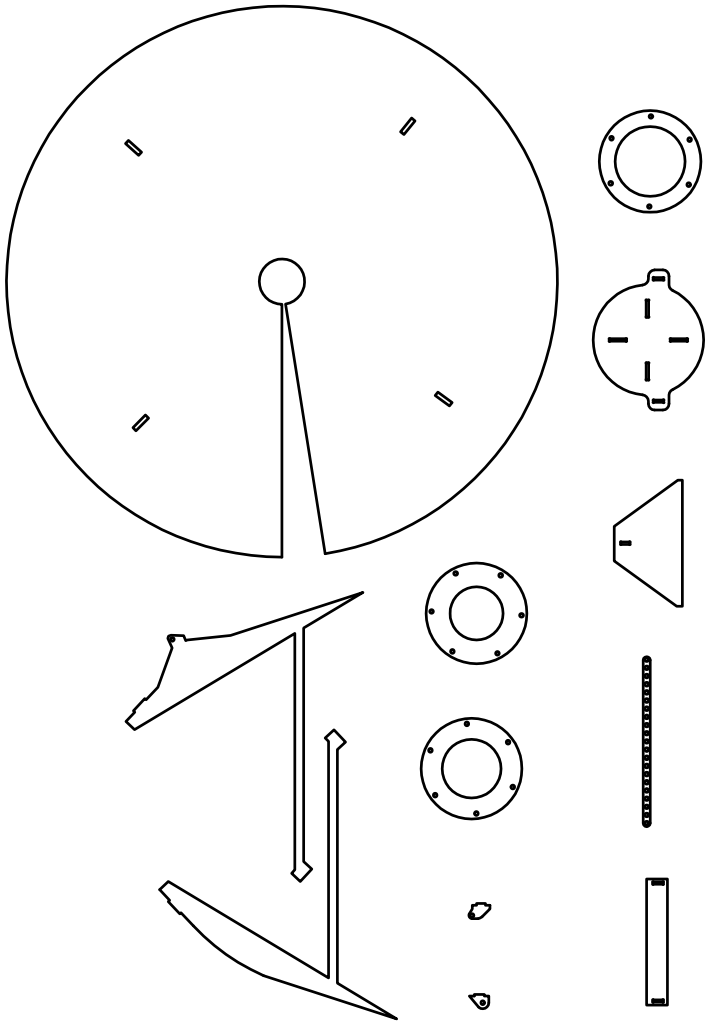


13	PED BASE	1	SA 300 Ø228 x 4
ITEM	BESKRYWING	AANTAL	MATERIAAL / SPESIFIKASIES

UNIVERSITEIT VAN STELLENBOSCH	SKAAL OP A 1:3	TITEL: PEDESTAL BASE	
	MATE IN mm		
STUDENTE No. 14596709	TEKENAAR BR POTGIETER	NAGESIEN	DATUM 29/09/08
VEL No. 13 VAN 14 VELLE	No. ANT - 12		

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

Laser Cutting Drawings



Appendix B

Testing Information

Table B.1: Equipment summary

Section	Model	Description	Serial No.
Excitation Mechanism	Bruel & Kjaer 4809	Electrodynamic Exciter	1061061
Transduction System	PCB 33B32 PCB 208A02	Piezo-Electrical Accelerometer Load Cell	See Table B.2 S/N 9190
Analyser	LMS SCADAS III	Digital Data Acquisition and Frequency Analyser	45052524

Table B.2: Accelerometer specifications

Accelerometer	Serial No.	Sensitivity (mV/g)	Bias Level (V)
1	S/N 18868	99.1	10.7
2	S/N 18863	105.2	10.7
3	S/N 18862	96.4	10.6
4	S/N 34758	98.2	11.5
5	S/N 34757	99.8	11.2
6	S/N 18869	104.0	10.7
7	S/N 18871	96.8	10.7
8	S/N 38402	98.7	11.6
9	S/N 18867	98.9	10.9
10	S/N 31631	101.5	11.5
11	S/N 31632	100.0	11.5
12	S/N 31630	99.9	11.5
13	S/N 31628	102.3	11.5
14	S/N 34759	103.6	11.6

Table B.3: Accelerometer coordinates

Channel	Accelerometer	x (mm)	y (mm)	z (mm)
1	Load Cell			
2	1	260	550	0
3	2	510	320	0
4	3	550	-260	0
5	4	330	-510	0
6	5	-260	-550	0
7	6	-510	-300	0
8	7	-550	260	0
9	8	-310	510	0
10	9	0	210	260
11	10	210	0	260
12	11	0	-210	260
13	12	-210	0	260
14	13	0	60	470
15	14	-60	0	470

Appendix C

Python Scripts

```
#GENESIS FOR eXPERIMENT 2:
#rEAD RELEVENT DATA

#Grid that we are interested in:
P = [31858, 31853, 28170, 28165, 28287,
    28282, 27936, 27931, 31038, 30674,
    31200, 30836, 30519, 30552]
    
#This note is to help find the Accelerometers order correct because
#Their grid numbers are not in asending or desending order
#-------------------------------------------------------------
#Note    Acc1 - 13      Acc2 - 12    Acc3 - 3    Acc4 - 2
#       Acc5 - 5       Acc6 - 4     Acc7 - 1    Acc8 - 0
#       Acc9 - 10      Acc10 - 8    Acc11 - 11  Acc12 - 9
#       Acc13 - 7      Acc14 - 6
#-------------------------------------------------------------
#-------------------------------------------------------------
#Note    0 - Acc8       1 - Acc7     2 - Acc4    3 - Acc3
#       4 - Acc6       5 - Acc5     6 - Acc14   7 - Acc13
#       8 - Acc10      9 - Acc12    10 - Acc9   11 - Acc11
#       12 - Acc2      13 - Acc1
#-------------------------------------------------------------

#Open file to be read:
#Displacements of modes
f = open('Dish_dsg00.pch', 'r')

Accno = size(P)        #number of Accelerometers
Mno = 8                #Number of Modes
k = 0
A = zeros((Mno*Accno,4))

for line in f:
    if(line[0] != '-' and line[0] != '$' ):
        b = asarray(line.split())
        a = int(b[0])
        for ii in arange(0, size(P)):
            if(a == P[ii]):
                xyz = b[2:5]
                A[k,0] = P[ii]
                A[k,1:4] = xyz
                k = k + 1

#-------------------------------------------------------------
#Open file for grid positions:
g = open('Dish.dat', 'r')

Loc = zeros((size(P),4))
K = 0
for line in g:
    c = line.strip()
    if(c[0] == 'G'):
        d = asarray(c.split())
        D = int(d[1])
        for ii in arange(0, size(P)):
            if(D == P[ii]):
```

```
57                    X = c[24:32]
58                    Y = c[32:40]
59                    Z = c[40:48]
60                    if(c[30] == '-'):
61                        X = c[24:30] + 'e' + c[30:32]
62                    if(c[38] == '-'):
63                        Y = c[32:38] + 'e' + c[38:40]
64                    if(c[46] == '-'):
65                        Z = c[40:46] + 'e' + c[46:48]
66                    XYZ = array([d[1], X, Y, Z])
67                    Loc[K,:] = XYZ
68                    K = K + 1
69
70  #_____________________________________________________________________________
    _______
71
72
73  from scipy.io import write_array
74  write_array("Modes_Gen.txt",A)
75  write_array("Grid_Loc2.txt",Loc)
```

```
#Open file to be read:
#Coherence Plots and
#FRF plots
#Phase plots

#_____________________________________________________________________#

from scipy.io import read_array
Acc = '14'
Coh = 'D:/Documents and Settings/Potty/Desktop/SKA/Experiment_2/Exp_2/Attempt_1/Coherence_1/Coh_A'+Acc+'.txt'
A = read_array(Coh)
FRF = 'D:/Documents and Settings/Potty/Desktop/SKA/Experiment_2/Exp_2/Attempt_1/FRF_1/FRF_A'+Acc+'.txt'
B = read_array(FRF)

for i in arange(0, size(B)/3):
    B[i,1] = (B[i,1]**2 + B[i,2]**2)**0.5
    B[i,2] = (arctan(B[i,2]/B[i,1]))*(180/pi)
    
freq = 13.9

#~ figure()
#~ plot(,range(10),'+')

figure()
subplot(3,1,1)
plot(A[:,0], A[:,1])
title('freq')
xlabel('Frequency (Hz)')
ylabel('Coherence')
#~ axis([0, 64, 0, 1])
grid()
show()

subplot(3,1,2)
plot(B[:,0], B[:,1])
xlabel('Frequency (Hz)')
ylabel('Magnitude')
#~ axis([0, 64, 0, 0.02])
grid()
show()

subplot(3,1,3)
plot(B[:,0], B[:,2])
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
#~ axis([0, 64, -95, 95])
grid()
show()
```

```
1  #Lab plots for experimental two with a directional option.
2  #Only using first 5 modes
3
4  #----------------------USER INPUTS-----------------------------
5  Area = 0            #0 for top half
6                      #1 for bottom half
7  Keep = 0            #0 show figure 1 save figure
8  Di = 0              #1 for only directions no magnitude
9  modes = 5           #Number of modes wanted
10 DSet = 1           #Which try should be read
11 #----------------------USER INPUTS-----------------------------
12
13 #-------------------------------------------------------------
14 #Note:          Try 1    Try 2
15 #       Mode 1: 13.93    13.82
16 #       Mode 2: 23.61    23.64
17 #       Mode 3: 26.15    26.21
18 #       Mode 4: 32.06    32.05
19 #       Mode 5: 42.13    42.14
20 #-------------------------------------------------------------
21
22 #Import abilaty to read arrays from txt files
23 from scipy.io import read_array
24 #Import interpolator for curve fits
25 import scipy.interpolate as SI
26
27 #_____________________________________________________________
28 #Functions for getting curve fit
29 def curvefit(x,y):
30     spline = SI.InterpolatedUnivariateSpline(x, y)
31     steps = 200
32     X = linspace(min(x), max(x), steps)
33     C = spline(X)
34
35     return (X,C)
36
37 #_____________________________________________________________
38 #Read in Grid locations
39
40 Grid = read_array(file("Grid_Loc2.txt"))
41
42 G1 = Grid[13, 1:4]              #Accelerometer 1
43 G2 = Grid[12, 1:4]              #Accelerometer 2
44 G3 = Grid[3, 1:4]               #Accelerometer 3
45 G4 = Grid[2, 1:4]               #Accelerometer 4
46 G5 = Grid[5, 1:4]               #Accelerometer 5
47 G6 = Grid[4, 1:4]               #Accelerometer 6
48 G7 = Grid[1, 1:4]               #Accelerometer 7
49 G8 = Grid[0, 1:4]               #Accelerometer 8
50 G9 = Grid[10, 1:4]              #Accelerometer 9
51 G10 = Grid[8, 1:4]              #Accelerometer 10
52 G11 = Grid[11, 1:4]             #Accelerometer 11
53 G12 = Grid[9, 1:4]              #Accelerometer 12
54 G13 = Grid[7, 1:4]              #Accelerometer 13
55 G14 = Grid[6, 1:4]              #Accelerometer 14
56
```

```
57 #_____________________________________________________________
58 #Read in grid points with out accelerometers:
59 #Note: Top = 2; Left = 0; Bot = 3; Right = 1
60
61 GnA = read_array(file("Grid_LocGrid.txt"))
62
63 GL = GnA[0, 1:4]              #Left Arm
64 GR = GnA[1, 1:4]              #Right Arm
65 GT = GnA[2, 1:4]              #Top Arm
66 GB = GnA[3, 1:4]              #Bottom Arm
67 #_____________________________________________________________
68 #Read Displacements from text files
69 #Seperate all Accel modes into there own arrays
70 #E1 = [mode1;mode2; ...;modeN]
71
72 if(DSet == 1):
73     Exp = read_array(file("Modes_Exp2T1.txt"))
74 if(DSet == 2):
75     Exp = read_array(file("Modes_Exp2T2.txt"))
76 #Make only directional:
77 if(Di == 1):
78     for i in arange(1,4):
79         for ii in arange(0,14*5):
80             if(Exp[ii,i] == 0):
81                 Exp[ii,i] = 0
82             else:
83                 Exp[ii,i] = Exp[ii,i]/abs(Exp[ii,i])
84
85 k1 = 0
86 k2 = 14          #number of accelerometers
87 MnE = modes          #number of modes read in in Read.py
88
89 Scale = 1
90
91 E1 = Exp[k1:MnE*k2:k2,1:4]*Scale
92 E2 = Exp[k1+1:MnE*k2:k2,1:4]*Scale
93 E3 = Exp[k1+2:MnE*k2:k2,1:4]*Scale
94 E4 = Exp[k1+3:MnE*k2:k2,1:4]*Scale
95 E5 = Exp[k1+4:MnE*k2:k2,1:4]*Scale
96 E6 = Exp[k1+5:MnE*k2:k2,1:4]*Scale
97 E7 = Exp[k1+6:MnE*k2:k2,1:4]*Scale
98 E8 = Exp[k1+7:MnE*k2:k2,1:4]*Scale
99 E9 = Exp[k1+8:MnE*k2:k2,1:4]*Scale
100 E10 = Exp[k1+9:MnE*k2:k2,1:4]*Scale
101 E11 = Exp[k1+10:MnE*k2:k2,1:4]*Scale
102 E12 = Exp[k1+11:MnE*k2:k2,1:4]*Scale
103 E13 = Exp[k1+12:MnE*k2:k2,1:4]*Scale
104 E14 = Exp[k1+13:MnE*k2:k2,1:4]*Scale
105
106 #_____________________________________________________________
107
108 #_____________________________________________________________
109 #Grid Positions of top and Bottom half of the Model
110
111 #xy of Accel on arms info and feed
112 xTop = array([G9[0], G10[0], G11[0], G12[0], G13[0], G14[0],
```

```
113             GL[0], GR[0], GT[0], GB[0]])
114 yTop = array([G9[1], G10[1], G11[1], G12[1], G13[1], G14[1],
115             GL[1], GR[1], GT[1], GB[1]])
116 
117 #xyz of Accel on Dish
118 xBot = array([G1[0], G2[0], G3[0], G4[0], G5[0],
119             G6[0], G7[0], G8[0], G1[0], G2[0]])
120 yBot = array([G1[1], G2[1], G3[1], G4[1], G5[1],
121             G6[1], G7[1], G8[1], G1[1], G2[1]])
122 zBot = array([G1[2], G2[2], G3[2], G4[2], G5[2],
123             G6[2], G7[2], G8[2], G1[2], G2[2]])
124 
125 #Labelling Plot
126 Labels = array(['Mode 1: 13.88 Hz', 'Mode 2: 23.62 Hz',
127                 'Mode 3: 26.18 Hz', 'Mode 4: 32.06 Hz',
128                 'Mode 5: 42.14 Hz', 'Mode 6: 42.90 Hz',
129                 'Mode 7: 45.76 Hz', 'Mode 8: 48.05 Hz'])
130 
131 #_____________________________________________________________
132 #Start Plotting:
133 
134 if(Area == 0):
135     #Label Accel positions
136     Name = array(['A9', 'A10', 'A11', 'A12', 'A13', 'A14',
137                 'GL', 'GR', 'GT', 'GB'])
138 
139     for i in arange(0, modes):
140         scale = 1
141 
142         dx = array([E9[i,0], E10[i,0], E11[i,0], E12[i,0],
143             E13[i,0], E14[i,0], 0, 0, 0, 0])
144 
145         dy = array([E9[i,1], E10[i,1], E11[i,1], E12[i,1],
146             E13[i,1], E14[i,1], 0, 0, 0, 0])
147 
148          #Scale factor:
149         scale = 1
150         if((GL[0]-GR[0])/4 < max(dx) or (GL[0]-GR[0])/4 < max(dy)):
151             scale = 10--->#Scale displacements
152         if(Di != 1):
153             if(i == modes-1):
154                 scale = 200
155             if(i == 0):
156                 scale = 2
157 
158         dx = dx/scale
159         dy = dy/scale
160         print(scale)
161 
162         Ex = xTop + dx              #new position in x
163         Ey = yTop + dy              #new position in y
164 
165         #Do curve fit between points:LEFT AND RIGHT------------
166         ELRx = array([Ex[7], Ex[1], Ex[5], Ex[3], Ex[6]])
167         ELRy = array([Ey[7], Ey[1], Ey[5], Ey[3], Ey[6]])
168         [X_LR, C_LR] = curvefit(ELRx, ELRy)
```

```
169         #LEFT AND RIGHT------------
170         
171         #Do curve fit between points:TOP AND BOTTOM------------
172         #For plotting x and y are swopped so GTBY is actually x values
173         ETBy = array([Ex[9], Ex[2], Ex[4], Ex[0], Ex[8]])
174         ETBx = array([Ey[9], Ey[2], Ey[4], Ey[0], Ey[8]])
175         [X_TB, C_TB] = curvefit(ETBx, ETBy)
176         #TOP AND BOTTOM------------
177         
178          #Plot
179         figure(i)
180         plot(xTop, yTop, '^b', Ex, Ey, 'or',
181              C_TB,X_TB,'--',X_LR,C_LR,'-.')
182         
183         #Label Points:
184         for ii in arange(0, size(dx)):
185             text(Ex[ii], Ey[ii], Name[ii])
186             text(xTop[ii], yTop[ii], Name[ii])
187         xlabel('x')
188         ylabel('y')
189         title(Labels[i])
190         legend(['Original','Displaced'])
191         axis([-0.475, 0.475, -0.475, 0.475])
192         grid()
193         if(Keep == 0):
194             show()
195         else:
196             #Save Figure
197             Save = "Exp2Top_M" + str(i+1) + "T" + str(DSet-2) +".eps"
198             savefig(Save)
199 #--------------------Top Half-----------------------------------
200 
201 #--------------------Bottom Half--------------------------------
202 if(Area == 1):
203     #Label Accel positions
204     Name = array(['A1', 'A2', 'A3', 'A4', 'A5',
205                   'A6', 'A7', 'A8', 'A1', 'A2'])
206  #_____________________Unwrap section:____________________________
207 
208     #Calculate angles between points
209     rad = average((xBot**2 + yBot**2)**0.5)         #Radius of Accel
210     dth = zeros(size(xBot)-1)     #Because one less angle than Accel
211 
212     for i in arange(0,size(dth)):
213         l = ((xBot[i+1] - xBot[i])**2 + (yBot[i+1] - yBot[i])**2)**0.5
214         dth[i] = arccos(l**2/(2*rad**2) - 1)*180/pi
215     dth = (180-dth)*pi/180
216 
217     #x values for unwrapped dish:
218     Xunw = zeros(size(xBot))
219     for i in arange(1,size(xBot)):
220         Xunw[i] = Xunw[i-1] + rad*dth[i-1]
221 #_____________________Unwrap section:____________________________
222 
223     #Now plot and put fit to x and z values:
224     for i in arange(0,modes):
```

```
225         dz = array([E1[i,2], E2[i,2], E3[i,2], E4[i,2], E5[i,2],
226                     E6[i,2], E7[i,2], E8[i,2], E1[i,2], E2[i,2]])
227 
228         Ez = zBot + dz                  #new position in z
229 
230         #Do curve fit between points:
231         [X_fit, C_fit] = curvefit(Xunw , Ez)
232 
233         #Plot
234         figure(i)
235         plot(Xunw, zBot, '^b', Xunw, Ez, 'or', X_fit, C_fit)
236         axis([-0.1, max(Xunw)+0.1, min(Ez)- 0.5, max(Ez)+0.5])
237          #Label Points:
238         for ii in arange(0, size(dz)):
239             text(Xunw[ii], Ez[ii], Name[ii])
240             text(Xunw[ii], zBot[ii], Name[ii])
241         xlabel('x')
242         ylabel('y')
243         title(Labels[i])
244         legend(['Original','Displaced'])
245         axis([-0.2, 4.5, -4.5, 6])
246         grid()
247 
248         if(Keep == 0):
249             show()
250         else:
251             #Save Figure
252             Save = "Exp2Bot_M" + str(i+1) + "T" + str(DSet-2) +".eps"
253             savefig(Save)
254 #---------------------Bottom Half---------------------------------
```

```
#GENESIS FOR eXPERIMENT 2:
#Plot displacements in two different 2D planes
#1 with the Dish acc and 1 with support and feed acc's:

#This note is to help find the Accelerometers order correct because
#Their grid numbers are not in asending or desending order
#-------------------------------------------------------------
#Note   Acc1 - 13      Acc2 - 12     Acc3 - 3     Acc4 - 2
#      Acc5 - 5       Acc6 - 4      Acc7 - 1     Acc8 - 0
#      Acc9 - 10      Acc10 - 8     Acc11 - 11   Acc12 - 9
#      Acc13 - 6      Acc14 - 7
#-------------------------------------------------------------
#-------------------------------------------------------------
#Note   0 - Acc8       1 - Acc7      2 - Acc4     3 - Acc3
#      4 - Acc6       5 - Acc5      6 - Acc13    7 - Acc14
#      8 - Acc10      9 - Acc12     10 - Acc9    11 - Acc11
#      12 - Acc2     13 - Acc1
#-------------------------------------------------------------

#----------------------USER INPUTS-----------------------------
Area = 0              #0 for top half
                      #1 for bottom half
Keep = 0              #0 show figure 1 save figure
modes = 5             #Number of modes wanted
#----------------------USER INPUTS-----------------------------

#Import abilaty to read arrays from txt files
from scipy.io import read_array
#Import interpolator for curve fits
import scipy.interpolate as SI

#_____________________________________________________________
#Functions for getting curve fit
def curvefit(x,y):
    spline = SI.InterpolatedUnivariateSpline(x, y)
    steps = 200
    X = linspace(min(x), max(x), steps)
    C = spline(X)

    return (X,C)
#_____________________________________________________________
#_____________________________________________________________
#Read in Grid locations

Grid = read_array(file("Grid_Loc2.txt"))

G1 = Grid[13, 1:4]              #Accelerometer 1
G2 = Grid[12, 1:4]              #Accelerometer 2
G3 = Grid[3, 1:4]               #Accelerometer 3
G4 = Grid[2, 1:4]               #Accelerometer 4
G5 = Grid[5, 1:4]               #Accelerometer 5
G6 = Grid[4, 1:4]               #Accelerometer 6
G7 = Grid[1, 1:4]               #Accelerometer 7
G8 = Grid[0, 1:4]               #Accelerometer 8
G9 = Grid[10, 1:4]              #Accelerometer 9
G10 = Grid[8, 1:4]              #Accelerometer 10
```

```
57 G11 = Grid[11, 1:4]              #Accelerometer 11
58 G12 = Grid[9, 1:4]               #Accelerometer 12
59 G13 = Grid[7, 1:4]               #Accelerometer 13
60 G14 = Grid[6, 1:4]               #Accelerometer 14
61
62 #____________________________________________________________
63 #Read Displacements from text files
64 #Seperate all Accel modes into there own arrays
65 #A1 = [mode1;mode2; ...;modeN]
66
67 File = 'Modes_Gen.txt'
68
69
70 A = read_array(file(File))
71
72 k1 = 0
73 k2 = 14          #number of accelerometers
74 Mno = 17         #number of modes read in in Read.py
75 A8 = A[k1:Mno*k2:k2,1:4]
76 A7 = A[k1+1:Mno*k2:k2,1:4]
77 A4 = A[k1+2:Mno*k2:k2,1:4]
78 A3 = A[k1+3:Mno*k2:k2,1:4]
79 A6 = A[k1+4:Mno*k2:k2,1:4]
80 A5 = A[k1+5:Mno*k2:k2,1:4]
81 A14 = A[k1+6:Mno*k2:k2,1:4]
82 A13 = A[k1+7:Mno*k2:k2,1:4]
83 A10 = A[k1+8:Mno*k2:k2,1:4]
84 A12 = A[k1+9:Mno*k2:k2,1:4]
85 A9 = A[k1+10:Mno*k2:k2,1:4]
86 A11 = A[k1+11:Mno*k2:k2,1:4]
87 A2 = A[k1+12:Mno*k2:k2,1:4]
88 A1 = A[k1+13:Mno*k2:k2,1:4]
89
90 #____________________________________________________________
91 #____________________________________________________________
92 #Read in grid points with out accelerometers:
93
94 GnA = read_array(file("Grid_LocGrid.txt"))
95
96 GL = GnA[0, 1:4]                #Left Arm
97 GR = GnA[1, 1:4]                #Right Arm
98 GT = GnA[2, 1:4]                #Top Arm
99 GB = GnA[3, 1:4]                #Bottom Arm
100
101
102 #____________________________________________________________
103 #____________________________________________________________
104 #Grid Positions of top and Bottom half of the Model
105
106 #xy of Accel on arms info and feed
107 xTop = array([G9[0], G10[0], G11[0], G12[0], G13[0], G14[0],
108             GL[0], GR[0], GT[0], GB[0]])
109 yTop = array([G9[1], G10[1], G11[1], G12[1], G13[1], G14[1],
110             GL[1], GR[1], GT[1], GB[1]])
111
112 #xyz of Accel on Dish
```

```
113 xBot = array([G1[0], G2[0], G3[0], G4[0], G5[0],
114               G6[0], G7[0], G8[0], G1[0], G2[0]])
115 yBot = array([G1[1], G2[1], G3[1], G4[1], G5[1],
116               G6[1], G7[1], G8[1], G1[1], G2[1]])
117 zBot = array([G1[2], G2[2], G3[2], G4[2], G5[2],
118               G6[2], G7[2], G8[2], G1[2], G2[2]])
119 
120 #For labeling of plots
121 #Label Accel positions
122 
123 #Labelling Plot
124 Labels = array(['Mode 1: 13.72 Hz', 'Mode 2: 24.62 Hz',
125         'Mode 3: 24.83 Hz', 'Mode 4: 34.62 Hz', 'Mode 5: 42.74 Hz',
126         'Mode 6: 48.73 Hz', 'Mode 7: 49.05 Hz', 'Mode 8: 49.13 Hz'])
127 
128 
129 
130 #_____________________________________________________________
131 #Start Plotting:
132 
133 #--------------------Top Half-----------------------------------
134 if(Area == 0):
135     #Label Accel positions
136     Name = array(['A9', 'A10', 'A11', 'A12', 'A13', 'A14',
137                   'GL', 'GR', 'GT', 'GB'])
138 
139     for i in arange(0, modes):
140 
141         dx = array([A9[i,0], A10[i,0], A11[i,0], A12[i,0],
142             A13[i,0], A14[i,0], 0, 0, 0, 0])
143 
144         dy = array([A9[i,1], A10[i,1], A11[i,1], A12[i,1],
145             A13[i,1], A14[i,1], 0, 0, 0, 0])
146 
147         #Scale factor:
148         scale = 1
149         if((GL[0]-GR[0])/4 < max(dx) or (GL[0]-GR[0])/4 < max(dy)):
150             scale = 15--------->#Scale displacements
151         dx = dx/scale
152         dy = dy/scale
153 
154         Gx = xTop + dx              #new position in x
155         Gy = yTop + dy              #new position in y
156 
157         #Do curve fit between points:LEFT AND RIGHT------------
158         GLRx = array([Gx[7], Gx[1], Gx[5], Gx[3], Gx[6]])
159         GLRy = array([Gy[7], Gy[1], Gy[5], Gy[3], Gy[6]])
160         [X_LR, C_LR] = curvefit(GLRx, GLRy)
161         #LEFT AND RIGHT------------
162 
163         #Do curve fit between points:TOP AND BOTTOM------------
164         #For plotting x and y are swopped so GTBY is actually x values
165         GTBy = array([Gx[9], Gx[2], Gx[4], Gx[0], Gx[8]])
166         GTBx = array([Gy[9], Gy[2], Gy[4], Gy[0], Gy[8]])
167         [X_TB, C_TB] = curvefit(GTBx, GTBy)
168         #TOP AND BOTTOM------------
```

```
169
170         #Plot
171         figure(i)
172         plot(xTop, yTop, '^b', Gx, Gy, 'or',
173             C_TB,X_TB,'--',X_LR,C_LR,'-.')
174
175         #Label Points:
176         for ii in arange(0, size(dx)):
177             text(Gx[ii], Gy[ii], Name[ii])
178             text(xTop[ii], yTop[ii], Name[ii])
179         xlabel('x')
180         ylabel('y')
181         title(Labels[i])
182         legend(['Original','Displaced'])
183         axis([-0.475, 0.475, -0.475, 0.475])
184         grid()
185         if(Keep == 0):
186             show()
187         else:
188             #Save Figure
189             Save = "GenTop_M" + str(i+1) + ".eps"
190             savefig(Save)
191 #--------------------Top Half-----------------------------------
192
193 #--------------------Bottom Half--------------------------------
194 if(Area == 1):
195     #Label Accel positions
196     Name = array(['A1', 'A2', 'A3', 'A4', 'A5',
197                   'A6', 'A7', 'A8', 'A1', 'A2'])
198  #_____________________Unwrap section:_____________________________
199
200     #Calculate angles between points
201     rad = average((xBot**2 + yBot**2)**0.5)          #Radius of Accel
202     dth = zeros(size(xBot)-1)    #Because one less angle than Accel
203
204     for i in arange(0,size(dth)):
205         l = ((xBot[i+1] - xBot[i])**2 + (yBot[i+1] - yBot[i])**2)**0.5
206         dth[i] = arccos(l**2/(2*rad**2) - 1)*180/pi
207     dth = (180-dth)*pi/180
208
209     #x values for unwrapped dish:
210     Xunw = zeros(size(xBot))
211     for i in arange(1,size(xBot)):
212         Xunw[i] = Xunw[i-1] + rad*dth[i-1]
213
214  #_____________________Unwrap section:_____________________________
215     #Now plot and put fit to x and z values:
216     for i in arange(0,modes):
217         dz = array([A1[i,2], A2[i,2], A3[i,2], A4[i,2], A5[i,2],
218                     A6[i,2], A7[i,2], A8[i,2], A1[i,2], A2[i,2]])
219
220         Gz = zBot + dz                  #new position in z
221
222         #Do curve fit between points:
223         [X_fit, C_fit] = curvefit(Xunw , Gz)
224
```

```
225         #Plot
226         figure(i)
227         plot(Xunw, zBot, '^b', Xunw, Gz, 'or', X_fit, C_fit)
228         axis([-0.1, max(Xunw)+0.1, min(Gz)- 0.5, max(Gz)+0.5])
229          #Label Points:
230         for ii in arange(0, size(dz)):
231             text(Xunw[ii], Gz[ii], Name[ii])
232             text(Xunw[ii], zBot[ii], Name[ii])
233         xlabel('x')
234         ylabel('y')
235         title(Labels[i])
236         legend(['Original','Displaced'])
237         #~ axis([-0.2, 4.5, -1.5, 2])
238         grid()
239 
240         if(Keep == 0):
241             show()
242         else:
243             #Save Figure
244             Save = "GenBot_M" + str(i+1) + ".eps"
245             savefig(Save)
246 #--------------------Bottom Half--------------------------------
247 
```

```
1  #MAC of Directional:
2  #MAC for Experiment 2. MAC for 2 tries and Genesis:
3
4  #----------------------USER INPUTS-----------------------------
5  Di = 0            #Di = 1 for only directional
6  Area = 2          #0 for bot and 1 for top 2 for all
7  Plot = 1          #1 for plot colour map
8  Keep = 0          #0 show figure 1 save figure
9  NoAcc = 14        #Number of accelerometers wanted to analyse
10 modes = 5         # can only use 5 or 8 modes
11 #----------------------USER INPUTS-----------------------------
12
13 from scipy.io import read_array
14
15 #Want all tries as well as Genesis Data so they can be all compared
16 if (modes == 5):
17     file = "Modes_5_All.txt"
18 if(modes == 8):
19     file = "Modes_All.txt"
20
21 Shapes = read_array(file) #this file has no editting just as is
22
23 #Make only directional:
24 if(Di == 1):
25     for i in arange(1,15):
26         for ii in arange(0,15):
27             Shapes[ii,i] = Shapes[ii,i]/abs(Shapes[ii,i])
28
29 if(Area == 1):
30     U1 = Shapes[0:modes,0:9]
31     U2 = Shapes[modes:2*modes,0:9]
32     UG = Shapes[2*modes:3*modes,0:9]
33 if(Area == 0):
34     U1 = zeros((modes,7))
35     U1[:,0] = Shapes[0:modes,0]
36     U1[0:modes, 1:7] = Shapes[0:modes, 9:15]
37     U2 = zeros((modes,7))
38     U2[:,0] = Shapes[modes:2*modes,0]
39     U2[0:modes, 1:7] = Shapes[modes:2*modes, 9:15]
40     UG = zeros((modes,7))
41     UG[:,0] = Shapes[2*modes:3*modes,0]
42     UG[0:modes, 1:7] = Shapes[2*modes:3*modes, 9:15]
43 if(Area == 2):
44     U1 = Shapes[0:modes,:]
45     U2 = Shapes[modes:2*modes,:]
46     UG = Shapes[2*modes:3*modes,:]
47
48
49 Freq1 = U1[:,0]
50 Freq2 = U2[:,0]
51 FreqG = UG[:,0]
52 #U1: [Freq 1, Mode 1
53 #     Freq 2, Mode 2]
54 #     ...
55 #-------------------------------------------------------------------------
56 #Compare Try's 1 and 2:
```

```
57
58 #Initialise
59 MACV12 = zeros((size(Freq1)+1, size(Freq2)+1))
60 MACV12[0,1:size(Freq2)+2] = Freq2
61 MACV12[1:size(Freq1)+2,0] = Freq1
62 #Try 1 rows and Try 2 Columns
63
64 for i in arange(0,size(Freq1)):
65     for ii in arange(0,size(Freq2)):
66         Mt = sum(U1[i,1:NoAcc+2]*U2[ii,1:NoAcc+2])      #Top part of equation
67         Mbl = sum(U1[i,1:NoAcc+2]*U1[i,1:NoAcc+2])      #Bottom Left of equation
68         Mbr = sum(U2[ii,1:NoAcc+2]*U2[ii,1:NoAcc+2])    #Bottom right of equation
69         MAC12 = Mt**2/(Mbl*Mbr)
70         MACV12[i+1,ii+1] = MAC12
71 #-------------------------------------------------------------------------------
   ----------
72 #Comparison with Genesis:
73
74 #Initialise
75 MACV1G = zeros((size(FreqG)+1, size(Freq1)+1))
76 MACV1G[0,1:size(Freq1)+2] = Freq1
77 MACV1G[1:size(FreqG)+2,0] = FreqG
78 #Gen rows and Try 1 Columns
79
80 MACV2G = zeros((size(FreqG)+1, size(Freq2)+1))
81 MACV2G[0,1:size(Freq2)+2] = Freq2
82 MACV2G[1:size(FreqG)+2,0] = FreqG
83 #Gen rows and Try 2 Columns
84
85 for i in arange(0,size(FreqG)):
86     for ii in arange(0,size(Freq1)):
87         Mt = sum(UG[i,1:NoAcc+2]*U1[ii,1:NoAcc+2])      #Top part of equation
88         Mbl = sum(UG[i,1:NoAcc+2]*UG[i,1:NoAcc+2])      #Bottom Left of equation
89         Mbr = sum(U1[ii,1:NoAcc+2]*U1[ii,1:NoAcc+2])    #Bottom right of equation
90         MAC1G = Mt**2/(Mbl*Mbr)
91         MACV1G[i+1,ii+1] = MAC1G
92
93 for i in arange(0,size(FreqG)):
94     for ii in arange(0,size(Freq2)):
95         Mt = sum(UG[i,1:NoAcc+2]*U2[ii,1:NoAcc+2])      #Top part of equation
96         Mbl = sum(UG[i,1:NoAcc+2]*UG[i,1:NoAcc+2])      #Bottom Left of equation
97         Mbr = sum(U2[ii,1:NoAcc+2]*U2[ii,1:NoAcc+2])    #Bottom right of equation
98         MAC2G = Mt**2/(Mbl*Mbr)
99         MACV2G[i+1,ii+1] = MAC2G
100
101 #-------------------------------------------------------------------------------
   ----------
```

```
102
103 #Plotting Colour map:
104 if(Plot == 1):
105     if(Di == 0):
106         tlDi = ''
107     else:
108         tlDi = ' Dir'
109     if(Area == 0):
110         tl = ' Top' + tlDi
111     if(Area == 1):
112         tl = ' Bottom'
113     if(Area == 2):
114         tl = ' Whole'
115
116     figure()
117     Mac12= MACV12[1:size(Freq1)+1,1:size(Freq2)+1]
118     cax = imshow(Mac12, origin= 'lower', interpolation='nearest',
119                  extent=[1,size(Freq2), 1,size(Freq1)], vmin = 0.5)
120     cbar = colorbar(cax)
121     #~ title('TRY 1 vs TRY 2 ' + tl)
122     xlabel('Run 2 Modes'), ylabel('Run 1 Modes')
123     grid()
124     show()
125
126     figure()
127     Mac1G= MACV1G[1:size(FreqG)+1,1:size(Freq1)+1]
128     cax = imshow(Mac1G, origin= 'lower', interpolation='nearest',
129                  extent=[1,size(Freq1), 1,size(FreqG)], vmin = 0.5)
130     cbar = colorbar(cax)
131     #~ title('Test vs Initial FEM')
132     xlabel('Test Modes'), ylabel('Initial FEM Modes')
133     grid()
134     show()
135     from scipy.io import write_array
136     write_array("MAC_initFEM_Test.txt",Mac1G)
137
138     figure()
139     Mac2G= MACV2G[1:size(FreqG)+1,1:size(Freq2)+1]
140     cax = imshow(Mac2G, origin= 'lower', interpolation='nearest',
141                  extent=[1,size(Freq2), 1,size(FreqG)], vmin = 0.5)
142     cbar = colorbar(cax)
143     title('Genesis vs TRY 2 ' + tl)
144     xlabel('Exp Modes'), ylabel('Genesis Modes')
145     grid()
146     show()
```

```
#Calculate Genesis's modeshapes.
#With A1-8 in Experiments z directions
#Check the correct number of frequencies and which frequencies
#that need to be read in from "Modes_Gen.txt"

#----------------------USER INPUTS-------------------------------
Keep = 1              #1 to write to file
Norm = 0              #Normalise modeshapes
#----------------------USER INPUTS-------------------------------

#-----------------------------------------------------------------
#Note   Acc1 - 13      Acc2 - 12     Acc3 - 3     Acc4 - 2
#      Acc5 - 5       Acc6 - 4      Acc7 - 1     Acc8 - 0
#      Acc9 - 10      Acc10 - 8     Acc11 - 11   Acc12 - 9
#      Acc13 - 7      Acc14 - 6
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#Note   0 - Acc8       1 - Acc7      2 - Acc4     3 - Acc3
#      4 - Acc6       5 - Acc5      6 - Acc14    7 - Acc13
#      8 - Acc10      9 - Acc12     10 - Acc9    11 - Acc11
#      12 - Acc2     13 - Acc1
#-----------------------------------------------------------------

from scipy.io import read_array

#Import Genesis so to make a model vector.
A = read_array('Modes_Gen.txt')
#Modes_Gen_NN is in Grid numeric order and not in Accel order
#A = mode 1: A8:x y z
#             A7:x y z
#                ....
#            A1:x y z
#    mode 2: A8:x y z

FreqG = array([13.72, 24.62, 24.83, 34.62,
               42.74, 48.73, 49.05, 49.13])


#Get Accelerometers in coorect order
# AA = [Freq1 A1x A1y A1z .... A14x A14y A14z
#       Freq2 A1x A1y A1z .... A14x A14y A14z..]

k2 = 14                 #number of accelerometers
Mno = size(FreqG)       #number of modes read in in Read.py

AA = zeros((Mno,3*14+1))
#For Accelerometers 1-8 get magnitude of 3 directions
for i in arange(0,size(FreqG)):
    AA[i,0] = FreqG[i]
    AA[i,1:4] = A[14*i+13,1:4]      #Move 1 from 14 to 1
    AA[i,4:7] = A[14*i+12,1:4]      #Move 2 from 06 to 2
    AA[i,7:10] = A[14*i+3,1:4]      #Move 3 from 07 to 3
    AA[i,10:13] = A[14*i+2,1:4]     #Move 4 from 05 to 4
    AA[i,13:16] = A[14*i+5,1:4]     #Move 5 from 01 to 5
    AA[i,16:19] = A[14*i+4,1:4]     #Move 6 from 02 to 6
    AA[i,19:22] = A[14*i+1,1:4]     #Move 7 from 03 to 7
```

```
57      AA[i,22:25] = A[14*i+0,1:4]       #Move 8 from 04 to 8
58      AA[i,25:28] = A[14*i+10,1:4]      #Move 9 from 12 to 9
59      AA[i,28:31] = A[14*i+8,1:4]       #Move 10 from 10 to 10
60      AA[i,31:34] = A[14*i+11,1:4]      #Move 11 from 13 to 11
61      AA[i,34:37] = A[14*i+9,1:4]       #Move 12 from 11 to 12
62      AA[i,37:40] = A[14*i+7,1:4]       #Move 13 from 08 to 13
63      AA[i,40:43] = A[14*i+6,1:4]       #Move 14 from 09 to 14
64
65  aa = zeros((size(FreqG),15))
66  for i in arange(0,size(FreqG)):
67      #Check which coordinate is needed:!!!!!!!!!!!!!!!!!!!!!!
68      aa[i,9] = AA[i, 3*8+1]            #Only X coordinate wanted
69      aa[i,10] = AA[i, 3*9+2]           #Only Y coordinate wanted
70      aa[i,11] = AA[i, 3*10+1]          #Only X coordinate wanted
71      aa[i,12] = AA[i, 3*11+2]          #Only Y coordinate wanted
72      aa[i,13] = AA[i, 3*12+1]          #Only X coordinate wanted
73      aa[i,14] = AA[i, 3*13+2]          #Only Y coordinate wanted
74
75  #for 1st 8 Accelerometers
76      for ii in arange(0,8):
77          aa[i,ii+1] = AA[i, 3*ii+3]#/cos(16*pi/180)
78
79
80      if(Norm == 1):
81          #Normalise vectors with max displacement for each mode
82          aa[i,:] = aa[i,:]/max(abs(aa[i,:]))
83      aa[i,0] = FreqG[i]
84
85  if(Keep == 1):
86      from scipy.io import write_array
87      write_array("ModeShapeG.txt",aa)
88      #~ write_array("Check_AA.txt",AA)
```

```
1  #~ Runs optimisation with different initial values
2
3  #Import necessary modules
4  import shutil
5  import subprocess
6  import os
7  from scipy.io import read_array
8  from scipy.io import write_array
9
10 # Open orginal .dat file to read from.
11 Org_f = 'check.dat'
12 read = open(Org_f,'r')
13
14 It = 0      #Initial run value starting at 0
15 No_run = 1  #Number of runs wanted
16
17 #Array for all values to be stored
18 count1 = zeros((No_run,1))
19
20 for ii in arange(0,No_run):
21     #----------------------Running analysis----------------------------
22     print('Make new dat files')
23     print('Run number: ',It)
24     #Copy file so new .dat with new initial values can be created
25     run = 'check_run_' + str(It) + '.dat'
26     shutil.copyfile(Org_f, run)
27
28     read = open(Org_f,'r')
29     write = open(run, 'w')
30
31     no_DVAR = 0                  #Counter for number of design variables
32     Ini = uniform(-0.004, 0.004, 21)  #random initial value generator
33     Ini[12] = uniform(0.0016, 0.002, 1)
34
35     #Read info out of orginal flie and write to new file.
36     for line in read:
37         c = asarray(line.split())
38         if(c[0] != 'DVAR'):
39             write.write(line)
40         #Write new initial values.
41         if(c[0] == 'DVAR'):
42             k = str(Ini[no_DVAR])
43             #Achieve correct format
44             if (Ini[no_DVAR] < 1e-4):
45                 k = '0.000000'
46             #~ print('k ', k)
47             k = k[0:8]
48             ln = line[0:24] + k + line[32:48]
49             write.write(ln)
50             write.write('\n')
51             no_DVAR = no_DVAR + 1
52
53
54     write.close()
55     read.close()
56     #Call genesis to do analysis.
```

```
57      program = 'D:\\Program Files\\vrand\\genesis10.1/bin/genesis101.exe ↵
    ' + run
58      subprocess.call(program)
59
60      #----------------------Reading from output file---------------------
61      file = 'check_run_' + str(It) + '.out'
62      output = open(file, 'r')
63      #open file for writing deign history to
64      DeHist = 'Design_History_' + str(It) + '.txt'
65      deshis = open(DeHist,'w')
66
67      #find design history in the out put file.
68      lookup = ' DESIGN    OBJECTIVE    MAXIMUM CONSTRAINT\n'
69      count = 0
70      for line in output:
71          #Count design varaibles
72          if ( line.strip().startswith('* D E S I G N') ) :
73              count = count + 1
74
75          if(line == lookup):
76              print('wooHoo', line)
77
78              #read lines from look up.
79              for line in output:
80                  #Only read from values.
81                  if ( line =='\n') :
82                      for line in output:
83                          c = asarray(line.split())
84                          deshis.write(line)
85                          #stop reading at end of design cycles
86                          if (c[0] == str(count/3-1)):
87                              stop = 1
88                              break
89                      if(stop == 1):
90                          break
91
92      output.close()
93      count1[ii] = count
94      It = It + 1
95
96  deshis.close()
97
98  #----------------------Take all history to 1 file----------------------
99
100 #all design cycles with all runs
101 all = zeros((sum(count1)/3,3))
102
103 for It in arange(0, No_run):
104     #read array from file
105     DeHist = 'Design_History_' + str(It) + '.txt'
106     A = read_array(DeHist)
107     cs = int(count1[It]/3)
108     ll = It*(cs)
109     all[ll:ll+cs,:] = A
110
111     #Remove unnecessary files
```

```
112         os.remove(DeHist)
113
114 #write all runs to one file.
115 write_array("Design_History.txt",all)
116
```

```
#Plot design history and/or constraint violation

#Import necessary modules
import shutil
import subprocess
import os
from scipy.io import read_array
from scipy.io import write_array

A = read_array('Design History.txt')



w = zeros((10,2)) #number of design runs in run
count = 0
for ii in arange(1,size(A)/3):
    if(A[ii,0] < A[ii-1,0]):
        w[count,1] = A[ii-1,0]
        w[count,0] = count
        count = count + 1
    
w[count,1] = A[ii,0]
w[count,0] = count
print('w ',w)

mi = zeros((10,2))
mi[:,0] = w[:,0]
st = 0
en = 2
mi[0,1] = min(A[st:en+1,1])
for ii in arange(1,10):
    st = 1 + en
    en = st + w[ii,1]
    mi[ii,1] = min(A[st:en+1,1])
    #~ print(st, en)
    
sorted = sort(mi, axis = 0)
a = list(sorted[:,1])
a.reverse()

figure()
plot(mi[:,0], mi[:,1], '-o')
xlabel('Iteration'), ylabel('Objective Function')
show()

figure()
plot(sorted[0:3,0], a[0:3], '-o',sorted[3:11,0], a[3:11],'-*')
plot(sorted[:,0], a)
legend(['Infeasible','Feasible'])
xlabel('Iteration'), ylabel('Objective Function')
show()

```